Aleksi Wallenius

Hammashoitokoneen keskitetty asetusten hallinta ja replikointi

Sähkötekniikan korkeakoulu

Diplomityö, joka on jätetty opinnäytteenä tarkastettavaksi diplomi-insinöörin tutkintoa varten Espoossa 7.3.2014.

Työn valvoja:

Prof. Raimo Sepponen

Työn ohjaaja:

DI Olli Kattelus

Aalto-yliopisto
Sähkötekniikan korkeakoulu

AALTO-YLIOPISTO
SÄHKÖTEKNIIKAN KORKEAKOULU

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Aleksi Wallenius

Työn nimi: Hammashoitokoneen keskitetty asetusten hallinta ja replikointi

Päivämäärä: 7.3.2014 Kieli: Suomi Sivumäärä:7+51

Elektroniikan laitos

Professuuri: Sovellettu elektroniikka Koodi: S-66

Valvoja: Prof. Raimo Sepponen

Ohjaaja: DI Olli Kattelus

Diplomityön tavoitteena oli suunnitella ja toteuttaa nykyaikaisen hammashoitokoneen asetusten hallinta ja replikointi -ominaisuus PC-ohjelmistolle. Ominaisuuden pohjana oli toiselle hoitokoneelle toteutettu vastaava ominaisuus. Ominaisuudella oli siis voitava ottaa talteen hoitokoneen asetukset tietokoneelle ja pystyttävä lähettämään asetukset takaisin yhdelle tai useammalle hoitokoneelle.
Työn alussa esitellään ominaisuuden toteuttamiseksi käytettyjä teknologioita ja niille löydettyjä vaihtoehtoja. Eri vaihtoehtoja vertaillaan ja perustellaan niiden välillä tehtyjä valintoja. Työssä kuvataan myös, kuinka ominaisuuden toimintaa suunniteltiin ja sen vaatimuksia pohdittiin. Joidenkin ominaisuuden osien toteutuksien valintoja selvitetään tarkemminkin.
Työn tavoitteessa onnistuttiin eli saatiin toteutettua toimiva ominaisuus, jonka toimintaperiaatetta ja käyttöä esitellään ja onnistumista arvioidaan työn loppuosassa. Lopuksi pohditaan vielä tapoja, joilla ominaisuuden toimintaa voitaisiin kehittää ja erityisesti, kuinka sen tietoturvaa voitaisiin parantaa.

Avainsanat: Asetusten hallinta, Hammashoitokone, HTTP, CGI, SQL, Java, C, C++, shell-skripti

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Aleksi Wallenius

Title: Centralized Settings Management And Replication of a Dental Unit

Date: 7.3.2014 Language: Finnish Number of pages:7+51

Department of Electronics

Professorship: Applied Electronics Code: S-66

Supervisor: Prof. Raimo Sepponen

Instructor: M.Sc. (Tech.) Olli Kattelus

The goal of this Master's Thesis was to design and to implement for a PC software the settings management and replication feature of a modern dental unit. As the basis for the feature was the corresponding feature implemented for another dental unit. So the feature had to be fit to be used to transfer and to save the settings of a dental unit to a computer and to send these settings back to one or more units. In the beginning of the Thesis technologies used to implement this feature and alternative techonologies found for them are presented. Different options are compared and choices made between them are justified. In the Thesis it's also described how the operation of the feature was planned and how its requirements were pondered. The choices of the implementations of some of the parts of the feature are clarified in even more detail.

The Thesis was successful in achieving its goal. This means that a functioning feature was implemented. Its operational principle and use is presented and displayed and its success is judged in the later parts of the thesis. Lastly ways to improve the functioning of the feature and especially its information security are pondered.

Keywords: Settings management, Dental unit, HTTP, CGI, SQL, Java, C, C++, shell script

Esipuhe

Ensimmäiseksi haluan kiittää Professori Raimo Sepposta työn valvonnasta ja Olli Kattelusta ohjauksesta ja molempia kärsivällisyydestä työn valmistumisaikataulun venyessä. Kiitokset myös Planmecalle ja planmecalaisille tuesta ja mahdollisuudesta työn tekoon, ja erityisesti Jari Kannonkerälle kannustuksesta.

Kiitän myös vanhempiani tuesta ja kaikkia kavereitani, erityisesti Matti Fouchault-Airasmaata ja Hanna Hoi Yee Tammista, jaksamisesta kuunnella valitustani diplomityön teon hankaluudesta. Kiitokset ansaitsevat myös Amândio Fondo, José Augusto Nhantumbo ja koko muu ASSCODECHA-järjestön väki Maputosta. He antoivat myöskin tukensa ja ennen kaikkea mahdollisuuden työn teolle varsinaisen työni ohella.

Lopuksi haluan mainita vielä Funky Monkeys -hostellin Etelä-Afrikan Nelspruitissa. Se tarjosi minulle sopivat puitteet kolmen viikon ajan joulun 2013 tienoilla tämän diplomityön valtaosan kirjoittamiselle.

Maputo, 16.2.2014

Aleksi Wallenius

Sisältö

Lyhenteet

ASP	Active Server Pages
CGI	Common Gateway Interface
FTP	File Transfer Protocol
FTPS	FTP Secure
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IP	Internet Protocol
IPSec	Internet Protocol Security
Java RMI	Java Remote Method Invocation
JSP	JavaServer Pages
MIME	Multipurpose Internet Mail Extensions
NFS	Network File System
RFC	Request for Comments
SCP	Secure Copy
SFTP	SSH File Transfer Protocol
SMTP	Simple Mail Transfer Protocol
SQL	Structured Query Language
SSH	Secure Shell
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TFTP	Trivial File Transfer Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
URL	Uniform Resource Locator
WWW	World Wide Web

1 Johdanto

Tämä diplomityö on tehty suomalaisessa hammaslääketieteen alan yrityksessä nimeltä Planmeca Oy. Työ tehtiin osana tekijän työtä Planmecan hammashoitokonedivisioonan ohjelmistotuotekehitysosastolla. Planmeca suunnittelee ja valmistaa hammashoitokoneita, hammasröntgenlaitteita ja näiden hallintaan tarkoitettuja työasemaohjelmistoja. Planmeca on Planmeca Group -yhtiöryhmän emoyhtiö. Yhtiöryhmään kuuluu lisäksi viisi muuta yhtiötä: Planmed Oy, Plandent Oy, LM-Instruments Oy, Opus Systemer AS ja Triangle Furniture Systems Inc. Näistä esimerkiksi Planmed suunnittelee, valmistaa ja markkinoi kuvantamislaitteita mammografiaa ja ortopedistä kuvantamista varten ja Plandent toimittaa laitteita, tarvikkeita ja palveluja hammasklinikoille ja -laboratorioille. Yhteensä yhtiöryhmä työllistää lähes 2500 henkilöä. [1] [2]

Hammashoitokoneella tarkoitetaan nykyaikaista laitekokonaisuutta joka sisältää hammaslääkärin potilastuolin lisäksi muun muassa lääkärin ja hoitajan tarvitsemat instrumentit ja oheislaitteet sekä ohjaus- ja puhdistusjärjestelmät. Sovereign on Planmecan kehittynein hammashoitokone ja siksi yhtiön lippulaivatuote. Sovereignin erikoisuutena on erityisesti sen korkea muokattavuuden aste. Hoitokone voidaan esimerkiksi helposti muuttaa oikeakätisen lääkärin käyttämästä vasemmankätiseksi. Modulaarisuutensa ansiosta Sovereign on hankittavissa lukuisilla erilaisilla kokoonpanoilla ja asennustavoilla. Kuvassa 1 näkyy Sovereign-hoitokone eräässä muodossaan. Myös hoitokoneen istuimen värit on valittavissa laajasta valikoimasta. [3] [4]

Toinen yhtiön myymistä hammashoitokoneista on nimeltään Planmeca Compact. Se on ominaisuuksiltaan jonkin verran Sovereignia riisutumpi. Planmecalla on myös PC:llä ajettava Romexis-ohjelmisto. Romexiksen käytetyimmät ominaisuudet liittyvät yrityksen kuvantamistuotteisiin. Ohjelmalla voidaan muun muassa hallita kuvattavana olleita potilaita ja tutkia heidän kuviaan. Romexikseen on kuitenkin saatavilla valinnainen Clinic Management -moduuli, jolla taas voidaan hallita esimerkiksi yhden, tai tarvittaessa jopa useamman, klinikan kaikkia hoitokoneita. Klinikkatasolta voidaan siirtyä myös yksittäisen koneen tarkkailuun ja jopa etäohjaamiseen. Hoitokoneiden käytöstä kerätään erilaisia tietoja, joita voi jälkikäteen tutkia Romexiksen lokikirjoista, kuvaajista ja erilaisista Romexiksen luomista raporteista. Tässä työssä tehty ominaisuus on osa Clinic Management -moduulia, joten jatkossa tässä työssä Romexis-termi käsittää ohjelman oletusominaisuuksien lisäksi myös Clinic Management -moduulin. Clinic Management -moduuli on oletuksena käytössä muun muassa yliopistoilla opetuskäytössä olevien hoitokoneiden yhteydessä. [5] [6] [7]

Romexis-ohjelmisto voidaan jakaa kolmeen itsenäisesti ajettavaan, mutta toistensa kanssa keskustelevaan, ohjelmaan. Romexis Server -palvelinohjelmia on kerrallaan käynnissä yhdessä työympäristössä tavallisesti yksi kappale. Palvelimella on tiedossa kaikki ympäristöstä löytyvät hoitokoneet. Sovereign-hoitokoneilla on tiedossa palvelimen IP-osoite ja ne ilmoittautuvat sille käynnistyessään lähettämällä viestin. Palvelin pyytää hoitokoneelta lisätietoja sen konfiguraatiosta ja tallentaa ne tietokantaansa. Yhteys Compactille muodostetaan hieman eri tavalla. Sen kohdalla yhteyden aloittaa Romexis. Palvelin muodostaa yhteyden itsenäisen ohjelmansa kautta.

Kuva 1: Sovereign-hoitokone. [3]

Ohjelma on nimeltään MultiProxy, ja sille on asetettu tiedot kaikkien Compactien IP-osoitteista. Kutakin Compactia kohden palvelin käynnistää Planlink-prosessin, joka hoitaa varsinaisen suoran keskusteluyhteyden hoitokoneeseen ja toimii eräänlaisena tulkkina hoitokoneen ja Romexis-ohjelmiston välillä. Kolmas Romexiksen pääosa on itse asiakasohjelma, Romexis Client. Se on ainoa osa, jolla on yleiselle käyttäjälle näkyvä graafinen käyttöliittymä. Tavallisesti asiakasohjelmia on käynnissä omansa kullekin hoitokoneelle ja niitä pyöritetään hoitokoneiden yhteydessä olevilla PC-koneilla. Client-ohjelma saa palvelimelta muun muassa tiedot eri hoitokoneista. Client voi ottaa myös oman yhteyden yksittäisiin hoitokoneisiin. Palvelimen tavoin Compactiin yhdistäessä asiakasohjelma käynnistää tätä varten PlanLink-prosessin. Lisäksi on olemassa Romexis Admin -ohjelma, jolla asetetaan eri asetuksia

Romexiksen käyttöä varten. Nämä sisältävät esimerkiksi MultiProxyn tarvitsemat Compactien tiedot. [5] [6] [7]

Tässä työssä toteutettiin Romexiksen Clinic Management -moduuliin Sovereignhammashoitokoneen asetusten hallinta ja replikointi -ominaisuus, jolla Sovereignin asetukset voidaan tallentaa Romexis-palvelimelle ja asettaa myöhemmin samat asetukset yhdelle tai useammalle hoitokoneelle etäyhteyden kautta. Ominaisuus oli omassa muodossaan toteutettu aiemmin Planmecan toiselle hoitokoneelle, Compactille. Sen pohjalta Sovereignin vastaavaa ominaisuutta lähdettiin kehittämään. Romexikseen haluttiin yhtenevät ominaisuudet kaikille eri hoitokoneille. Ominaisuuden vaatimuksia mietittiin paljon, mutta lopulta päädyttiin tekemään ensin ulkoisesti täysin Compactia vastaava toiminnallisuus, jota voidaan sitten tarvittaessa laajentaa tai muuttaa.

Ominaisuuden käytöstä Compactilla ei ole hirveästi tietoa, mutta sillä on selkeästi potentiaalista hyötyä. Sitä on myös toivottu myyntiosaston puolelta kentältä saadun palautteen perusteella. Sovereign on suhteellisen uusi tuote, eikä sitä ole vielä myyty suuria määriä, joten ominaisuuden tarjoamat hyödyt eivät ole vielä olleet suuressa tarpeessa. Suurilla klinikoilla, joilla lääkärit käyttävät useita eri hoitokoneita ja kutakin hoitokonetta käyttää useampi lääkäri, on huomattavaa apua siitä, että lääkäri voi aina kullekin työpisteelle tullessaan kirjautua tunnuksillaan Romexikseen ja asettaa hoitokoneeseen haluamansa asetukset. Hoidon lopetettuaan lääkäri voi taas tallentaa halutessaan mahdolliset tekemänsä muutokset, jotka ovat sitten käytettävissä kaikilla klinikan koneilla. Samantapaista toiminnallisuutta on saatu aikaan myös Sovereignin ominaisuudella, jolla lääkäri voi tuoda omat asetuksensa hoitokoneelle USB-muistitikulla ja myös tallentaa sinne tekemänsä muutokset. Tämän ominaisuuden käyttö ei kuitenkaan ole niin kätevää, eikä se ole läheskään yhtä kattava ja monikäyttöinen.

Toisaalta ominaisuudesta voi olla suurta apua yliopistoilla hammaslääketieteellisissä opetusluokissa, joissa on kymmeniä jos ei jopa satoja hammashoitokoneita yhdessä tilassa. Niissä usein halutaan asettaa kaikkien oppilaiden hoitokoneet samaan alkutilaan. Työssä toteutetun ominaisuuden avulla siitä tulee vaivatonta myös Sovereignille. Opettajan tai teknikoiden ei tarvitse yksi kerrallaan käydä läpi kymmeniä hoitokoneita asettaakseen ne kaikki samaan tilaan.

Kolmas selkeä käyttökohde ominaisuudelle on asennusten helpottaminen. Tästä löytyikin omakohtaista kokemusta yrityksen sisällä. Nopeuttaa huomattavasti työtä, jos hoitokoneet voidaan kaikki helposti kerralla asettaa asiakkaan haluamaan tilaan ennen luovutusta. Vaikka ominaisuuden käytöstä tai sen toivomisesta ei saatu suoraa palautetta, on sen tuoma hyöty säästetyssä ajassa ja käytön helpottamisessa kuitenkin niin selkeä, että ominaisuuden tarpeellisuus on kiistaton. Se on myös selkeä myyntivaltti yritykselle, jonka kilpailijoilla ei vielä tätä ominaisuuttaa vastaavaa toiminnallisuutta tai edes Clinic Management -moduuliin tapaista ohjelmaa ole tarjolla.

2 Kirjallisuuskatsaus käytetyistä ja vaihtoehtoisista teknologioista

2.1 TCP/IP-protokollaperhe

Internetin protokollaperhettä kutsutaan TCP/IP-protokollaperheeksi sen ensimmäisten määriteltyjen ja yleisimmin käytettyjen protokollien mukaan. Näiden lisäksi perhe sisältää lukuisia muita protokollia, joiden pohjalla itse internet rakentuu. Protokollat ryhmitellään neljälle eri abstraktiotasolle, jotka ovat alimmasta alkaen: linkkitaso (Link layer), internet-taso (Internet layer), siirtotaso (Transport layer) ja sovellustaso (Application layer). Linkkitasoon kuuluvat teknologiat, jotka liittyvät verkon yhden pisteen viestintään. Internet-tasolta löytyvät taas yksittäiset verkot yhdistävät teknologiat. Siirtotaso hoitaa prosessien välisen kommunikaation ja sovellustasolla on ohjelmat, jotka hoitavat tukitoimintoja ja ovat yhteydessä käyttäjään. [8] [9] [10]

2.1.1 Internet Protocol

Internet Protocol eli IP on internet-tasoon kuuluva pakettien välittämiseen tietoverkkojen sisällä ja välillä käytetty kommunikaatioprotokolla. Se on monella tapaa internetin tärkein protokolla ollen käytännössä sen pohja ja mahdollistaen sen koko olemassaolon. IP:n avulla paketit kulkevat kohteisiinsa paketteihin määriteltyjen IP-osoitteiden mukaan. Yleisin käytössä oleva IP-versio on neljä. Versioon kuusi ollaan kuitenkin siirtymässä, koska yksilölliset IP-osoitteet ovat käytännössä loppuneet version neljä tavasta määrittää osoiteavaruus. [8] [9] [11] [12]

2.1.2 Transmission Control Protocol

Tranmission Control Protocol, TCP, on siirtotason protokolla, joka tarjoaa mahdollisuuden lähettää dataa sovellusten välillä. Protokolla on luotettava, mikä tarkoittaa sitä, että se sisältää toiminnallisuuksia, jotka pyrkivät estämään viestien katoamisen matkalla. Se myös pyrkii korjaamaan virheitä lähetetyssä datassa. TCP on myös yhteydellinen protokolla eli siinä osapuolten välille avataan yhteys, jolloin jokaisen paketin ei tarvitse sisältää tietoa tarkoitetusta vastaanottajasta ja jokainen viesti tulee vastaanottajalle samassa järjestyksessä kuin ne lähetettiin. [8] [9] [13] [14]

2.1.3 User Datagram Protocol

User Datagram Protocol, UDP, on TCP:n tavoin siirtotason protokolla. Myös sen avulla voidaan lähettää viestejä ohjelmien välillä. UDP ei kuitenkaan ole luotettava protokolla eli siinä ei ole mitään virheenkorjausta tai viestin perillemenon varmistusta. Se on myös yhteydetön protokolla, missä viestit vain lähetetään kukin erikseen vastaanottajan osoitteeseen. Sitä voi siis käyttää tarkoituksiin, joissa tällaiset hienoudet eivät ole välttämättömiä tai ne hoidetaan sovelluksien toimesta. Hyvänä puolena saadaan tällöin kevyempi ja nopeampi protokolla sellaista vaativiin käyttökohteisiin. [8] [9] [15] [16]

2.2 Terminaalisovellukset

Työssä käytetään terminaalisovellusta nimeltä Telnet. Sen avulla saadaan tekstipohjainen yhteys Sovereignin ACCU-kortin Linux-pohjaiseen käyttöjärjestelmään. Telnet on tietoturvaltaan hyvin puutteellinen. Telnetin onkin korvannut monissa nykyaikaisissa sovelluksissa usein Secure Shell eli SSH-sovellus. SSH tarjoaa hyvin suojatun ja salatun kirjautumisen, datasiirron ja terminaaliyhteyden käyttäjälle. Komentorivin etäkäytön lisäksi sitä voidaan käyttää salaamaan muuten suojaamatonta internet-yhteyttä käyttävän protokollan viestiliikenne. [8] [17] [18] [19] [20]

2.3 Tiedonsiirto

2.3.1 Trivial File Transfer Protocol

Trivial File Transfer Protocol eli TFTP on nimensä mukaisesti hyvin yksinkertainen tiedostonsiirtomenetelmä ja tarvitsee vain vähän muistia. Siksi sitä käytetäänkin yleensä automatisoituihin toimintoihin kuten käynnistystiedoston siirtämiseen koneiden välillä paikallisesti. TFTP käyttää yleensä UDP:tä tiedonsiirtoon, mutta se voidaan toteuttaa käyttäen myös jotain muuta siirtoprotokollaa. Protokolla toimii siten, että lähettäjä lähettää dataa vakiosuuruisissa erissä, ja vastaanottaja kuittaa kunkin paketin saapuneeksi. Tarvittaessa sama paketti lähetetään uudestaan. Myös kuittaus voidaan lähettää uudestaan, jos seuraavaa pakettia ei kuuulu.

TFTP on hyvin riisuttu verrattuna esimerkiksi FTP:hen (File Transfer Protocol). Sillä voi lähinnä vain siirtää tiedostoja. Siinä ei esimerkiksi ole mahdollisuutta suojata yhteyttä kirjautumisella tai kysellä saatavilla olevia tiedostoja. Ensimmäisen ominaisuuden takia sitä käytetään lähinnä paikallisverkoissa. Myös hoitokoneita käytetään tavallisesti paikallisverkoissa, joten TFTP olisi sinänsä ollut riittävä tämän työn käyttöön. TFTP-yhteys oli myös jo valmiiksi osittain toteutettu Sovereignin ja Romexiksen välille. Toki mahdottomuus suojata yhteyttä millään tavalla, olisi heikentänyt hoitokoneen tietoturvaa. [8] [21] [22]

2.3.2 File Transfer Protocol

File Transfer Protocol eli FTP on TFTP:n tavoin osa TCP/IP-Internet-protokollaperhettä. Se on huomattavasti TFTP:tä monipuolisempi. Se toimii selkeästi asiakasohjelma-palvelin-arkkitehtuurimallin mukaisesti. FTP on toteutettu toimimaan TCP:n päällä. Siinä on muun muassa toteutettu käyttäjätunnusten ja salasanojen vaatiminen. Alkuperäisessä muodossaan FTP käyttää kahta yhtäaikaista TCP-yhteyttä. Toisella liikkuu varsinainen data ja toisella hoidetaan yhteyden ohjaus. Sama ohjausyhteys säilyy koko session ajan ja sen katketessa koko sessio päättyy. Sen sijaan datayhteyksiä luodaan aina uusi kullekin datansiirrolle. FTP:stä on kuitenkin olemassa uudempia versioita, joissa käytetään HTTP:n tavoin vain yhtä yhteyttä.

FTP:n puutteena on edelleen se, että se ei olla millään tavalla salattu. Ulkopuolisella, jolla on pääsy FTP-palvelimen ja asiakasohjelman väliseen viestintään, on mahdollisuus saada tietoonsa selkokielisenä muun muassa käytetty käyttäjätunnus ja salasana. FTP:tä onkin yhdistetty muihin teknologioihin paikkaamaan tämä puu-

te. FTPS eli FTP Secure yhdistää FTP:hen TLS:n ja SSL:n, jolloin koko yhteys on salattu, eikä kirjautumistietoja tai siirretyn datan sisältöä pysty ulkopuoliset saamaan selville. Toinen vaihtoehto on nimeltään FTP over SSH. Siinä nimensä mukaisesti FTP-yhteydet putkitetaan SSH-yhteyden kautta. Ongelmaksi tässä lähestymisessä muodostuu helposti se, että FTP käyttää useampia yhteyksiä. Ohjausyhteys salataan, mutta avatessa uusi datayhteys SSH-asiakasohjelma ei ymmärrä salata myös sitä. Siksi onkin käytettävä SSH-asiakasohjelmia, joissa tähän ongelmaan on kiinnitetty erityistä huomiota. [8] [9] [23] [24] [25] [26] [27] [28]

2.3.3 Secure Copy ja SSH File Transfer Protocol

Secure Copy eli SCP on SSH:ta käyttävä tiedostonsiirtoprotokolla. SSH:tä käytetään kirjautumiseen ja yhteyden salaamiseen. SCP on siis suojattu protokolla. SCP toimii TCP:n kautta. SCP on sinänsä melko riisuttu teknologia sen mahdollistaessa ainoastaan tiedostojen siirron. Sitä ei ole määritelty missään RFC:ssä. SSH File Transfer Protocol eli SFTP sen sijaan sisältää FTP:n tavoin hieman enemmän ominaisuuksia. SFTP mahdollistaa tiedostojen siirtämisen lisäksi niiden käytön ja hallinnan yhteyden yli. Useimmiten myös sitä käytetään SSH:n kautta, mutta sitä voidaan periaatteessa käyttää missä tahansa suojatussa yhteydessä kuten TLS- tai VPN-yhteydessä. Se olettaa, että käytetyn teknologian yhteys on salattu ja se on jo hoitanut käyttäjän autentikoinnin. SFTP vaatii pohjalleen luotettavan datavirran. Vaikka SFTP muistuttaakin FTP:tä ominaisuuksiltaan, se ei ole sama asia kuin SSH:n yli käytetty FTP, vaan aivan oma protokollansa. [29] [30]

2.3.4 Network File System

Vielä yksi tiedostonsiirtomenetelmä, joka olisi ollut mahdollisesti sopiva työn käyttötarkoitukseen on nimeltään Network File System eli NFS. Se ei varsinaisesti ole tiedonsiirtomenetelmä vaan jaettu tiedostojärjestelmä. Sen avulla fyysisesti toisella koneella oleva data voidaan saada osaksi toisen koneen tiedostojärjestelmää. Ulkoisesti käyttäjälle siis näyttää, että kaikki data sijaitsee samalla koneella. Näin asetustiedot olisi voitu jakaa suoraan Romexis-palvelimelta Sovereignin käyttöön, ja Sovereign olisi voinut tallentaa asetuksensa suoraan palvelimelle. Tällä teknologialla, kuten mahdollisesti muillakin vaihtoehtoisilla teknologioilla, ongelmiksi toteutuksessa olisivat saattaneet muodostua Javan tuki kyseiselle teknologialle tai mahdollisesti tarvittavien kirjastojen löytäminen sekä niiden alustariippuvuus. Romexista voidaan käyttää useammissa eri käyttöjärjestelmissä, mutta NFS on yleisimmin käytössä vain Linux-pohjaisissa käyttöjärjestelmissä. Toki vastaavanlaisia jaettuja tiedostojärjestelmiä on muitakin. [8] [31] [32]

2.4 Hypertext Transport Protocol

Hypertext Transport Protocol eli HTTP on protokolla, johon World Wide Web perustuu. Sitä käyttäen web-palvelimet (server) ja web-selaimet tai muut asiakasohjelmat (client) keskustelevat keskenään. HTTP-protokolla perustuu pyyntöihin (request) ja vastauksiin (response). Pyynnön lähettäjänä voi olla esimerkiksi PC:n

Kuva 2: Esimerkit työssä käytetyistä URL-osoitteista.

```
http://192.168.1.100/cgi-bin/download_user.cgi
http://192.168.1.100/cgi-bin/download_config.cgi
http://192.168.1.100/cgi-bin/upload_user.cgi
http://192.168.1.100/cgi-bin/upload_config.cgi
```

web-selain, joka pyytää vaikka käyttäjänsä valitsemaa verkkosivua, kuvaa tai muuta resurssia. Selain kertoo pyynnössä URL-osoitteen (Uniform Resource Locator) avulla haluamansa resurssin. Palvelin lähettää sitten pyynnön mukaisen vastauksen. Yleisimmin HTTP:tä käytetään siis web-sivujen siirtämiseen, mutta sitä voidaan käyttää myös muihin tarkoituksiin, kuten esimerkiksi tämän työn tavoin tiedostojen siirtämiseen.

URL koostuu useasta osasta, joista kukin voidaan kerrallaan jättää pois tietyissä tilanteissa. Kuvassa 2 on esitetty esimerkit URL:listä tässä työssä tehtyihin ja käytettyihin CGI-skripteihin. URL:liä käytetään internetissä moniin tarkoituksiin. Tästä syystä URL:n alussa kerrotaan käytetty protokolla. Tämän työn tapauksessa URL alkaa siis kirjaimilla `http`. Jos käytettäisiin salattua HTTP-protokollaa, alku olisi `https`. Tämän jälkeen tulee halutun palvelimen osoite. Yleensä HTTP:n tapauksessa käytetään palvelimen domain-osoitetta IP-osoitteen sijaan, koska domain-osoite pysyy muuttumattomana vaikka koneen IP-osoite muuttuisikin esimerkiksi fyysisen palvelimen vaihtuessa. Tässä työssä hoitokoneella ei ole omaa domainia, vaan siihen viitataan suoraan Romexiksen tiedossa olevalla IP-osoitteella. Palvelimen osoitteen jälkeen voidaan antaa kaksoispisteen jälkeen käytettäväksi haluttu palvelimen portti. Jos porttia ei ole määritetty käytetään protokollan mukaista oletusporttia, joka HTTP:n tapauksessa on 80. Seuraavana URL:ssä on polku haluttuun resurssiin palvelimella. Polku on virtuaalinen, eli se ei välttämättä vastaa resurssin sijaintia palvelimen todellisessa hakemistorakenteessa. Sen sijaan palvelimen asetuksissa on määritetty, mihin todelliseen kansioon kukin URL-polku, jos mihinkään, viittaa. Usein on tapana, kuten tässäkin tapauksessa, että CGI-skriptit sijoitetaan kansioon, johon viitataan URL:ssä nimellä `cgi-bin` tai `cgi`, ja palvelin asetetaan hakemaan skriptiä oikeasta kansiosta. Palvelin tällöin myös ymmärtää ajaa skriptin sen sijaan, että palauttaisi vain sen sisällön. Muita URL:n osia ei tässä työssä käytetä. [33] [34] [8]

HTTP-pyynnöt ja vastaukset ovat rakenteeltaan hyvin samankaltaisia. Ne koostuvat otsikko-osasta (header) ja, tarvittaessa, varsinaisesta rungosta (body). Pyynnön otsikko-osa sisältää ensimmäisellä rivillä (request line) pyynnön tyypin, valitun resurssin URL-osoitteen ja käytetyn protokollan tyypin ja version. Muut rivit sisältävät lisätietoja pyynnöstä, esimerkiksi käyttäjän todennustiedot ja rungon sisällön tyypin ja koon. Vastaus alkaa tilarivillä (status line), joka kertoo jälleen käytetyn protokollan ja sen version, kolminumeroisen tilakoodin, joka kertoo kertoo pyynnön onnistumisesta tai mahdollisista virheistä tai erikoistilanteista, sekä kyseisen koodin merkityksen tekstimuodossa. Tilarivin jälkeen myös vastaus sisältää rivejä, jotka antavat lisätietoja vastauksesta. Otsikkorivien jälkeen on tyhjällä rivillä erotettu-

Kuva 3: Esimerkki työssä käytetystä GET-tyyppisestä HTTP-pyynnöstä.

```
GET /cgi-bin/download_config.cgi HTTP/1.1
Authorization: Basic cG1zb3ZlcmVpZ246c292ZXJlaWducG0=
Host: 192.168.1.100
Connection: Keep-Alive
User-Agent: Apache-HttpClient/4.2.1 (java 1.5)
[\r][\n]
```

Kuva 4: Esimerkki työssä käytetystä POST-tyyppisestä HTTP-pyynnöstä.

```
POST /cgi-bin/upload_config.cgi HTTP/1.1
Authorization: Basic cG1zb3ZlcmVpZ246c292ZXJlaWducG0=
Content-Length: 28864
Content-Type:  multipart/form-data;
boundary=3pZTf_KePS4LlFCkCAGoXFOhpY-4F6
Host: 192.168.1.100
Connection: Keep-Alive
User-Agent: Apache-HttpClient/4.2.1 (java 1.5)
--3pZTf_KePS4LlFCkCAGoXFOhpY-4F6
Content-Disposition:  form-data;
>> name="userfile";
>> filename="unitconf.tar"
Content-Type: application/octet-stream
[\r][\n]
Tiedoston sisältö
[\r][\n]
--3pZTf_KePS4LlFCkCAGoXFOhpY-4F6--
```

na mahdollisesti viestin runko. Pyynnöillä tämä voi sisältää esimerkiksi palvelimelle lähetettäviä lomaketietoja tai lähetettävän tiedoston. Vastauksen rungossa taas on sitten yleensä pyydetty resurssi, joka on tavallisimmin web-sivun sisältö HTML-muodossa. Tässä työssä vastaus sisältää osassa tapauksista pyydetyt asetustiedostot.

HTTP-pyyntöjä on siis erityyppisiä. Tässä työssä tarvitaan vain GET- ja POST-tyyppisiä pyyntöjä. Näistä on esimerkit kuvissa 3 ja 4. GET-pyynnöllä pyydetään palvelimelta jotakin resurssia esimerkiksi web-sivun sisältöä HTML-muodossa (Hypertext Markup Language). POST-pyynnöllä taas pyydetään palvelinta muokkaamaan palvelimella olevaa tietoa tai tekemään jotain pyynnön mukana lähetetylle datalle. Eri pyyntötyyppejä on listattuna taulukossa 1. Kuten kuvien 3 ja 4 esimerkeistä nähdään pakollisen ensimmäisen rivin jälkeen otsikossa on riveittäin avain-arvo-pareja, joissa alussa oleva avain on erotettu arvosta kaksoispisteellä. Taulukossa

Taulukko 1: HTTP-pyyntötyyppejä. [34]

Tyyppi	Kuvaus
GET	Pyytää palvelimelta annettua resurssia
HEAD	Käytetään kuten GET-pyyntöä, mutta palvelin palauttaa vain otsikkorivit ilman sisältöä
POST	Pyytää muuttamaan palvelimella olevaa tietoa
PUT	Pyytää palvelinta luomaan tai muuttamaan palvelimella olevaa resurssia
DELETE	Pyytää palvelinta poistamaan palvelimella olevan resurssin
CONNECT	Käytetään sallimaan suojattujen SSL-yhteyksien välittäminen HTTP-yhteyden kautta
OPTIONS	Pyytää palvelinta listaamaan mahdolliset pyyntötyypit annetulle resurssille
TRACE	Pyytää palvelinta toistamaan saamansa pyynnön otsikkorivit

Taulukko 2: HTTP-pyyntöjen yleisimpiä ja tässä työssä tärkeimpiä otsikkokenttiä. [34]

Otsikko	Kuvaus
Host	Kertoo palvelimen nimen tai IP-osoitteen
Content-Length	Kertoo pyynnön sisällön koon kahdeksan bitin tavuissa
Content-Type	Kertoo pyynnön sisällön tyypin
Authorization	Kertoo autentikoinnin tyypin ja käyttäjän käyttäjänimen ja salasanan salattuna
User-Agent	Kertoo käytetyn asiakasohjelman nimen, version ja alustan
Content-Disposition	Käytetään esimerkiksi lähetettäessä lomaketietoja POST-pyynnöllä kertomaan kenttien nimet ja mahdollisen lähetettävän tiedoston nimi

2 on esitelty yleisimpien ja tämän työn kannalta tärkeimpien otsikkorivien merkityksiä.

Kuvan 3 esimerkki on hyvin yksinkertainen. Siinä pyydetään `download_config.cgi`-skriptiä palvelimelta, jonka IP-osoite on 192.168.1.100. Palvelin tunnistaa, että kyseessä on skripti ja ajaa sen skriptin sisällön palauttamisen sijaan. `Authorization`-kentässä kerrotaan tunnistautumisen tavaksi `Basic` ja lähetetään käyttäjätunnus ja salasana salattuina. Tämä kenttä lähetetään yleensä vasta, kun palvelin on vastannut tilakoodilla 401, joka ilmoittaa, että kirjautuminen on tarpeen. Tässä työssä kenttä lähetetään kuitenkin välittömästi pakotettuna ensimmäisen pyynnön mukana, koska tuntemattomasta syystä yhteyttä ei muuten saatu muodostettua.

Kuvan 4 esimerkissä taas on POST-tyyppinen pyyntö, jolla lähetetään asetustiedosto hoitokoneelle. Pyynnön `Content-Type`-kentässä kerrotaan, että sisällön tyyppi on `multipart/form-data`, joka tarkoittaa moniosaista esimerkiksi HTML-lomakkeiden lähettämää dataa. Kukin osa on erotettu toisistaan `Boundary`-merkkisarjalla. Lisäksi jokaisen osan sisältö on kuvattava sitä ennen vähintään `Content-Type`-kentällä. Esimerkin tapauksessa `application/octet-stream` tarkoittaa osan sisällön olevan geneeristä bittivirtaa ilman tarkempaa määrittelyä. Se luonnollisesti muodostaa lähetettävän tiedoston sisällön. `Content-Disposition`-kentällä kerrotaan lisätietoja osasta. Tässä tapauksessa kyseessä on lomakekentän nimeltä `userfile` sisältö.

Kuva 5: Esimerkki työssä saadusta HTTP-vastauksesta GET-pyyntöön.

```
HTTP/1.1 200 OK
Date: Wed, 23 Oct 2013 14:10:01 GMT
Server: Apache/2.2.17 (Unix)
Last-Modified: Tue, 30 Oct 2012 14:31:14 GMT
ETag: "29981d-7000-4cd47a35d1080"
Accept-Ranges: bytes
Content-Length: 28672
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-disposition: attachment;
>> filename="unitconf-sovereign_3-2012-10-30_143114-5028.tar"
Content-type: application/x-tar
[\r][\n]
Tiedoston sisältö
```

Kuva 6: Esimerkki työssä saadusta HTTP-vastauksesta POST-pyyntöön.

```
HTTP/1.1 200 OK
Date: Thu, 24 Oct 2013 14:42:54 GMT
Server: Apache/2.2.17 (Unix)
Vary: Accept-Encoding
Content-Length: 30097
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html
[\r][\n]
<html>
Sivun ja mahdollisen tiedoston sisältö
</html>
```

`Filename`-parametrin ollessa läsnä tiedetään kyseessä olevan tiedosto. Parametrin arvo on tiedostolle ehdotettu oletustiedostonimi. Esimerkissä runko sisältää vain yhden osan. Rungon koon on vastattava `Content-Length`-kentän arvoa tavuina.

Kuvissa 5 ja 6 on esimerkit työssä mahdollisesti saatavista HTTP-vastauksista. Monet vastauksen otsikko-osan kentistä on samoja kuin pyynnöillä. Taulukossa 3 on esitelty lisänä joitain erityisesti HTTP-vastauksissa esiintyviä kenttiä. `WWW-Authenticate`-kenttää käytetään tilakoodin 401 kanssa kertomaan lisätietoja halutusta autorisoinnista. Tässä työssä sitä ei siis tarvita, koska, kuten sanottua, autorisointitiedot lähetetään aina automaattisesti ilman palvelimen pyyntöä.

Vastauksen ensimmäisen rivin tilakoodi on tyypillisimmin onnistuneen vastauk-

Taulukko 3: HTTP-vastausten yleisimpiä ja tässä työssä tärkeimpiä otsikkokenttiä. [34]

Otsikko	Kuvaus
Date	Kertoo vastauksen lähettämisen päivämäärän ja kellonajan
Last-Modified	Kertoo, milloin pyydetty resurssi on viimeksi muuttunut
Server	Kertoo palvelimen ohjelman, sen version ja käytetyn alustan
ETag	Kertoo pyydetyn resurssin yksilöivän tunnuksen
WWW-Authenticate	Kertoo asiakasohjelmalle lisätietoja palvelimen vaatimasta käyttäjän todennuksesta yhdessä tilakoodin 401 kanssa

sen tapauksessa 200. Näin tämänkin työn tapauksessa. Vastauskoodi 200 tarkoittaa, että vastaus pyyntöön on onnistunut ja sisältää jonkinlaisen runko-osan. Tätä käytetään lähinnä GET- ja POST-tyyppisten pyyntöjen vastauksissa. Erilaisia vastauksen tilakoodeja on lukuisia. Tilakoodit ovat aina kolminumeroisia ja ne voidaan jakaa ensimmäisen numeron mukaan viiteen ryhmään. Numerolla yksi alkavat ovat alemman tason tilakoodeja ja vastaus saattaa näissä tapaukissa sisältää pelkän tilarivin. Numerolla kaksi alkavat ovat kaikki erilaisia onnistuneeseen pyynnön toteuttamiseen viittaavia vastauksia. Kolmialkuiset koodit viittaavat siihen, että asiakasohjelman on tehtävä uusi pyyntö johtuen esimerkiksi resurssin siirtymisestä toiseen paikkaan. Neljäalkuinen koodi on merkki virheestä asiakasohjelman päässä. Lopulta vitosella alkava tila kertoo virheestä palvelimella.

Kuvan 5 GET-pyynnön vastauksessa on runkona hoitokoneen asetustiedostot pakattuna tar-tiedostoon. `Content-Disposition`-kentässä kerrotaan, että sisältö on attachment-tyyppiä. Selaimessa tämä aiheuttaa yleensä sen, että se ehdottaa oletuksena tiedoston tallentamista eikä yritä avata sitä selaimen ikkunaan. Oletustiedostonimen selain saa samaisen kentän `filename`-parametrista. `Content-Type`-kertoo tiedoston olevan tyypiltään `application/x-tar`. Tämä tarkoittaa, että kyseessä on tar-tyyppinen tiedosto, minkä voi toki nähdä myös tiedostopäätteestä. `Content-Type`-kentän arvoksi voi periaatteessa asettaa mitä tahansa, mutta on kuitenkin määritelty lista yleisesti hyväksyttyjä tunnuksia eri tiedostotyypeille. Näitä kutsutaan Internet Media Typeiksi. Ne kehitettiin alunperin kuvaamaan sähköpostien liitetiedostojen tyyppejä kehitettäessä nykyisin yleisesti käytössä olevaa, SMTP:tä (Simple Mail Transfer Protocol) käyttävää, ei-tekstimuotoiset liitetiedostot mahdollistavaa sähköpostistandardia eli MIME:ä (Multipurpose Internet Mail Extensions). Nykyisin tyyppejä käytetään muissakin yhteyksissä kuten HTTP-otsikoissa. [35] [36]

Kuvan 6 POST-pyynnön vastauksessa taas palautetaan HTML-muotoista tekstiä. Tätä saatetaan tulevaisuudessa tarvita kehitettäessä hoitokoneen web-käyttöliittymää eteenpäin. Yhdessä kohdassa HTML-sivua näytetään lähetetyn tiedoston sisältö tekstimuodossa. Jos tiedostoa ei ole lähetetty, vastaus-HTML sisältää lomakkeen, jonka avulla tiedoston voi lähettää. Tulevaisuuteen varautumisen lisäksi tämä auttoi ominaisuuden testaamisessa kehitysvaiheessa. [34] [8] [9] [37] [38]

2.5 TLS, SSL ja IPsec

Secure Sockets Layer, SSL, on itse asiassa Transport Layer Securityn, TLS, edeltäjä, mutta niistä käytetään usein yhteisnimitystä SSL/TLS. TLS (ja sitä ennen SSL) on salausprotokolla, jonka tarkoitus on tuoda tietoturvaa internetin yhteyksiin, jotta sitä ei voida salakuunnella. Nimensä mukaisesti sitä käytetään internetin siirtotasolla. Sillä voidaan salata esimerkiksi HTTP-protokollan viestiliikenne. Tällaisesta yhteydestä käytetään nimitystä HTTP Secure eli HTTPS. TLS:ää voidaan käyttää myös lukuisten muiden protokollien pohjalla varmistamaan tietoturvallisuus. Internet Protocol Security eli IPsec TLS:stä poiketen toimii sitä alemmalla eli internet-tasolla. Se salaa internetin IP-protokollan viestinnän. [8] [9] [39] [40] [41]

2.6 Common Gateway Interface

World Wibe Webin alkuaikoina suuri osa sen sisällöstä oli staattista eli se ei muuttunut pyynnöstä toiseen. Tämä saattoi tarkoittaa vaikkapa HTML:ää sisältävää tekstitiedostoa, joka oli siis valmiiksi olemassa ja joka palautettiin sellaisenaan. Nopeasti tuli kuitenkin tarpeelliseksi dynaamisen sisällön luominen. Sisältöä piti pystyä luomaan reaaliaikaisesti. Esimerkiksi käyttäjän syötteisiin kuten HTTP-pyynnön mukana lähetettyihin täytetyn lomakkeen tietoihin piti pystyä reagoimaan. Myös mahdollista sivulla olevaa tietoa oli pystyttävä päivittämään jokaisella latauksella vaikkapa tietokannasta.

Yksi ensimmäisistä tavoista tuottaa Webiin dynaamisesti sisältöä oli Common Gateway Interface eli CGI. CGI:lle on tullut lukuisia kilpailevia teknologioita, jotka ratkaisevat dynaamisen sisällön ongelman kukin omalla tavallaan, mutta ne eivät ole kuitenkaan täysin korvanneet CGI:tä sen ollessa vielä yleisesti käytössä. Näitä muita teknologioita selvitetään lyhyesti luvussa 2.6.1.

CGI ei ole ohjelmointikieli vaan nimensä mukaisesti rajapinta. Se on rajapinta web-palvelimen ja sen ajaman CGI-skriptin välillä. Skriptin voi siis toteuttaa lukuisilla eri kielillä. Yksi yleisimmin käytössä olevista on Perl-kieli. Perl-kielellä on monia hyviä puolia. Sitä ei muun muassa tarvitse kääntää, vaan koodi tulkitaan ajonaikaisesti. Tässä työssä käytetään C++-kieltä. Rajapinnan käsittelyä helpottamaan on monille kielille tehty CGI-kirjastoja. Perlille on olemassa ainakin CGI.pm-kirjasto ja tässä työssä käytetään C++:n CGICC-kirjastoa.

Yleisin web-palvelin on tässäkin työssä käytetty Apache, mutta myös useimmat muut web-palvelimet tukevat CGI:tä valmiiksi. Jotta CGI-skriptejä voidaan käyttää, on tätä varten kuitenkin asetettava oikein tiettyjä asioita web-palvelimen asetuksista. Kun palvelin saa HTTP-pyynnön (yleensä tyypiltään GET) jotakin resurssia varten, on palvelimen osattava päätellä, tuleeko resurssin sisältö palauttaa vai, kuten CGI:n tapauksessa, ajaa haluttu skripti. Palvelimen asetuksiin on säädettävä, mitkä tiedostot ovat ajettavia. Yleensä asetetaan, että CGI-tiedostot ovat esimerkiksi `/cgi-` tai `/cgi-bin`-kansiossa olevat tiedostot. Lisäksi voidaan ohjata ajettavaksi kaikki tiedostot, joiden tiedostopääte on .cgi. Näiden asetusten tekeminen on tietenkin erilaista eri palvelimilla, eikä sitä tässä käydä läpi.

Kun web-palvelin käynnistää CGI-skriptin, se tekee sen uudessa prosessissa. Se

antaa sille tietoja vakiosyötteen (standard input, STDIN) ja ympäristömuuttujien kautta. Skriptillä on siten käytössään monen muun asian lisäksi muun muassa koko saadun HTTP-pyynnön sisältö vaikkapa lomaketietoineen. Skripti tulostaa vakioulostuloon (standard output, STDOUT) rivit, jotka se haluaa palvelimen palauttavan asiakasohjelmalle. Tämä on usein HTML-koodia. Skripti voi myös ennen sitä määritellä HTTP-otsikkokenttiä. Palvelin saattaa vielä lisätä omia otsikkorivejään vastaukseen ennen sen palauttamista. [34]

2.6.1 Vaihtoehtoja CGI:lle

CGI:lle on kehitetty lukuisia vaihtoehtoja, jotka yrittävät ratkaista saman ongelman eri tavoin. Monet niistä pyrkii välttämään uuden prosessin luomisen dynaamista sisältöä luodessa, mikä on yksi CGI:n heikkouksista. Active Server Pages eli ASP mahdollistaa koodin kirjoittamisen HTML-tiedoston sisälle. Kielivaihtoehtoja on useita, mutta yleisin käytetty on Visual Basic. PHP taas on itsessään oma kielensä, jota palvelin tulkitsee ja suorittaa. Sitäkin voidaan kirjoittaa HTML:n sekaan tai omiin tiedostoihinsa. Sitä käytetään Apachen web-palvelimen kanssa.

Java servletit on teknologia, joka muistuttaa CGI:tä siinä on erillisiä skriptejä, jotka luovat dokumentteja. Kielenä käytetään aina Javaa ja rajapinta eroaa selkeästi CGI:stä. Java Server Pages eli JSP käyttää myös Javaa, mutta siinä koodia kirjoitetaan ASP:n tavoin HTML:n sekaan. FastCGI ja mod_perl taas ovat teknologioita, jotka käyttävät CGI-rajapintaa, mutta niissä käytetään pelkästään Perl-kieltä. Niissä tavallisen CGI:n uusien prosessien luomisesta johtuva hitaus on vältetty ensimmäisessä pitämällä Perl-prosessia koko ajan käynnissä palvelimen rinnalla ja toisessa integroimalla Perl-tulkki suoraan palvelimeen. Myös JavaScript-kieltä voidaan pitää ainakin osassa käyttötarkoituksista CGI:n korvaajana. Siinä lähestymistapa on kuitenkin toinen, sillä koodi suoritetaan vasta selaimen päässä. [34]

2.6.2 GNU CGICC -kirjasto

CGICC on ANSI-yhteensopivalle C++-kielelle toteutettu kirjasto, joka helpottaa CGI-rajapinnan käyttöä ja CGI-ohjelmien tekoa. Sen avulla voidaan muun muassa käsitellä helposti HTTP-POST- ja GET-pyyntöjen sisältämiä lomaketietoja. Se tukee myös tiedostojen lähettämistä HTTP:n kautta. Kirjasto sisältää runsaasti apuluokkia ja metodeja, joiden avulla voidaan esimerkiksi tuottaa kätevästi HTML-dokumentteja ja HTTP-vastauksen muita osia. CGICC on myös yhteensopiva FastCGI:n kanssa. [42]

2.7 Muita käytettyjä kirjastoja

2.7.1 Java Remote Method Invocation

Java Remote Method Invocation eli Java RMI on kirjasto ja ohjelmointirajapinta, jonka avulla Java-ohjelmat voivat kutsua metodeja ja käyttää olioita, jotka sijaitsevat toisessa Javan virtuaalikoneessa mahdollisesti toisella tietokoneella. Tyypillisessä RMI-kokoonpanossa on palvelin- ja asiakasohjelma. Palvelimella määritellään

RMI-rajapinnat ja ne toteuttavat oliot, joita voidaan sitten käyttää rajapinnan yli. RMI-nimirekisteriin on tallennettava RMI-oliot, jolloin asiakasohjelma voi hakea viittauksen haluamaansa olioon rekisteristä. Asiakasohjelma voi saada viittauksen etäolioon myös metodin argumenttien tai palautusarvon kautta. [43]

3 Ominaisuuden lähtökohdat ja suunnittelu

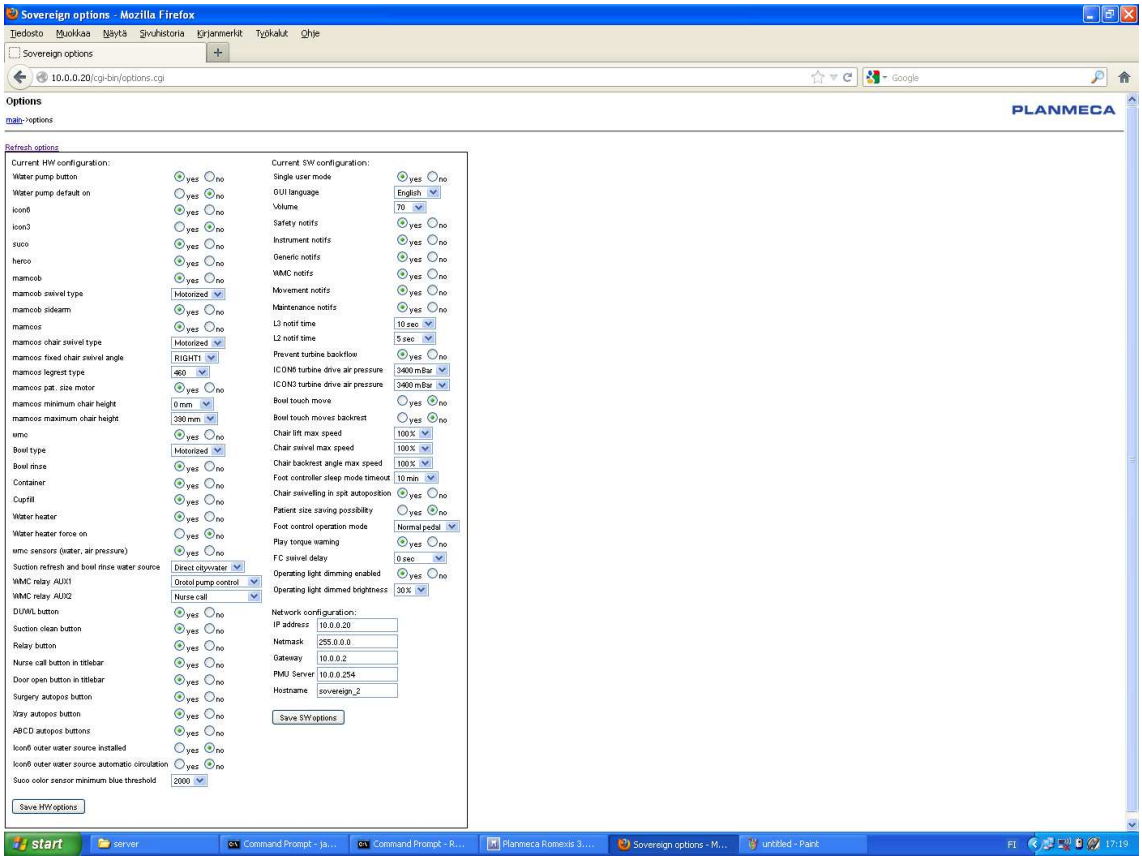
3.1 Ominaisuus Planmeca Compact -hoitokoneella

Tämän työn ominaisuuden toteutuksen lähtökohtana oli siis Planmecan toiselle hoitokonetyypille, Compactille, toteutettu vastaava ominaisuus. Compactilla sen asetukset on tallennettuna bitteinä suoraan muistiin tekstimuotoisten tiedostojen sijaan. Compact lähettää Romexikselle muiden tietojen mukana asetuksensa jatkuvasti tasaisin väliajoin. Asetusbitit tallennetaan muuttujaan, jonka sisältö tallennetaan kantaan käyttäjän niin halutessa. Asetusten tallennukseen käytetään kahta eri tietokantataulua. Päätauluun tallentuu yksi rivi per haettu asetus. Käyttäjäkohtaiset ja konekohtaiset asetukset tallentuvat erilleen. Aputauluun tallennetaan sitten useampia arvoja kustakin tallennetusta asetuksesta. Tallennettavia asioita ovat viite samaan aikaan tallennettuun toiseen asetukseen, eli viite käyttäjäkohtaisista asetuksista konekohtaisiin, asetukset tallentanut Romexis-käyttäjä, tieto asetusten globaaliudesta, tallennusnimi ja itse asetusbitit, jotka on eroteltu useammalle riville, koska yhden rivin datakentän maksimikoko on 2000 merkkiä. Bitit on siis koodattu tekstiksi kymmenkantaisena.

Vastaavasti asetuksia Compactille lähetettäessä lähetetään asetusbitit tietyssä järjestyksessä pienemmissä erissä Compactin viestiliikenneprotokollan mukaisesti muun viestiliikenteen seassa tietyissä paikoissa viestikehystä. Compact sitten kirjoittaa bitit muistiinsa oikealle alueelle. Compactilla kokemusta ominaisuuden käytöstä on lähinnä koneiden asennuksen yhteydestä, jolloin koneet on voitu helposti asettaa samaan alkutilaan asennuksen päätteeksi. Yleinen mielipide on kuitenkin selkeästi sellainen, että ominaisuudella olisi paljon enemmänkin potentiaalia, ja sitä pidetään hyödyllisenä lisänä ja hyvänä kilpailuvalttina kilpailijoihin nähden.

3.2 Sovereignin web-käyttöliittymä

Sovereignille on tehty selaimella käytettävä käyttöliittymä, jolla voi tehdä muutoksia hoitokoneen yleisempiin konekohtaisiin asetuksiin, huoltoasetuksiin, moottorin ajoasetuksiin ja joihinkin instrumenttiasetuksiin. Kuvissa 7 ja 8 näkyy lomakkeet, joilla voidaan muuttaa yleisiä konekohtaisia asetuksia ja instrumenttiasetuksia. Sivut luodaan Sovereignin HTTP-palvelimella Perl-kielellä kirjoitetuilla CGI-skripteillä. Kun lomakkeilla tehdään muutoksia asetuksiin, CGI-skriptit käsittelevät muutokset ja tekevät vastaavat muutokset oikeisiin tiedostoihin. Asetustiedostojen siirtoon käytettävät skriptit tehtiin myös CGI-rajapintaa hyödyntäen, jotta ne voitaisiin myöhemmin helposti integroida olemassaolevaan hallintajärjestelmään. Selaimella voidaan jo oikealla URL:llä hakea asetukset ja saada näkyviin lomake, jolla lähettää hoitokoneelle tiedostoja. Tiedostoja ei kuitenkaan vielä käsitellä mitenkään. CGI-skripti pitäisi muokata keskustelemaan Sovereignin pääohjelman kanssa ja ilmoittamaan saapuneista asetustiedostoista. Myös käyttöliittymän pääsivulle pitäisi lisätä linkit näihin CGI-skripteihin.



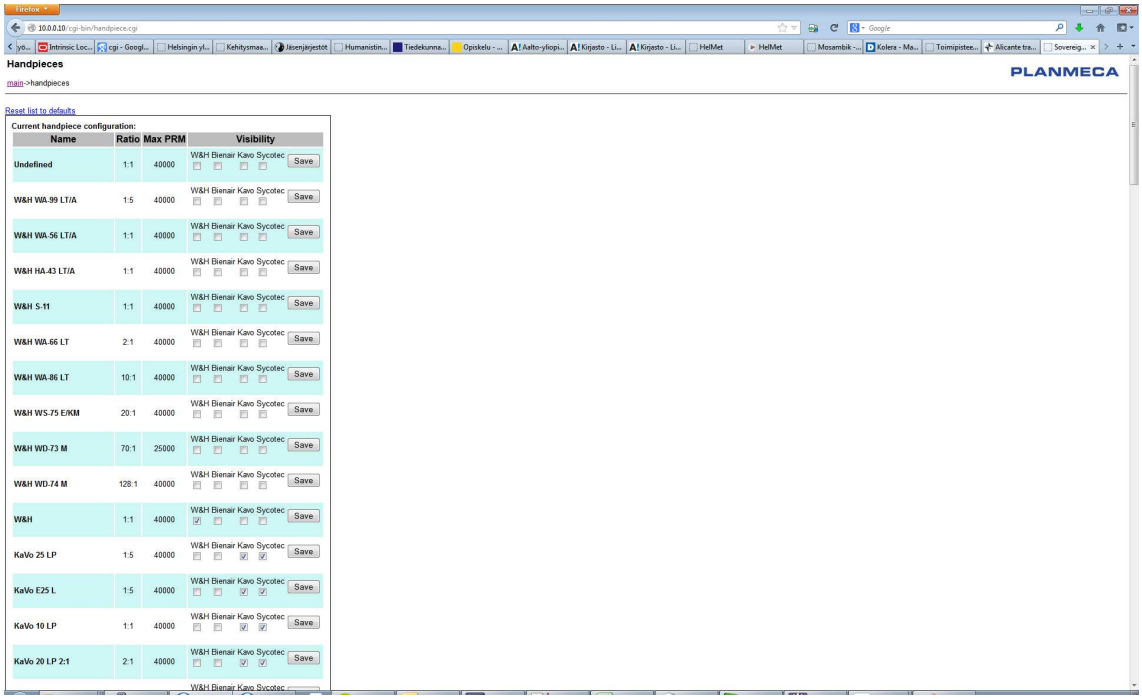
Kuva 7: Sovereignin web-käyttöliittymän konekohtaisten asetusten muokkaussivu.

3.3 Sovereignin ACCU-kortin arkkitehtuuri ja sisäinen viestintä

Sovereignin pääkortti on nimeltään ACCU. Sillä pyörii siis Sovereignin C++- ja C-kielillä toteutettu pääohjelmisto sekä myöskin työssä hyödynnetty HTTP-palvelin. Se on ainoa Sovereignin piirikorteista, joilla on tämän työn kannalta merkitystä. ACCU:n ohjelmisto pyörii useassa erillisessä itsenäisessä prosessissa, jotka keskustelevat toistensa kanssa käyttäen yhteistä kirjastoa. Kullekin prosessille on määritetty oma viestirajapintansa. Tärkein prosesseista on nimeltään Operator, joka toimii muiden prosessien välissä ja jolla on myös useita aliprosesseja. ACCU:n prosessit ovat yhteydessä myös hoitokoneen muiden laitteiden ohjelmistoihin CAN-väylän kautta. Tärkein ACCU:n prosesseista tämän työn kannalta on kuitenkin nimeltään Planlink. Se on hyvin itsenäinen prosessi, joka on kyllä yhteydessä Operatoriin, mutta hoitaa yksin keskustelun Romexiksen kanssa. Se pystyy hallitsemaan samanaikaisesti kymmenen yhteyttä eri Romexiksen osiin ja säikeisiin. [44] [45] [46] [47]

3.4 Romexis-arkkitehtuuri

Romexis-ohjelmisto koostuu siis käytännössä kolmesta osasta. Ne ovat Romexis-palvelin, Romexis-asiakasohjelma ja Romexis Planlink, jota voidaan kutsua myös



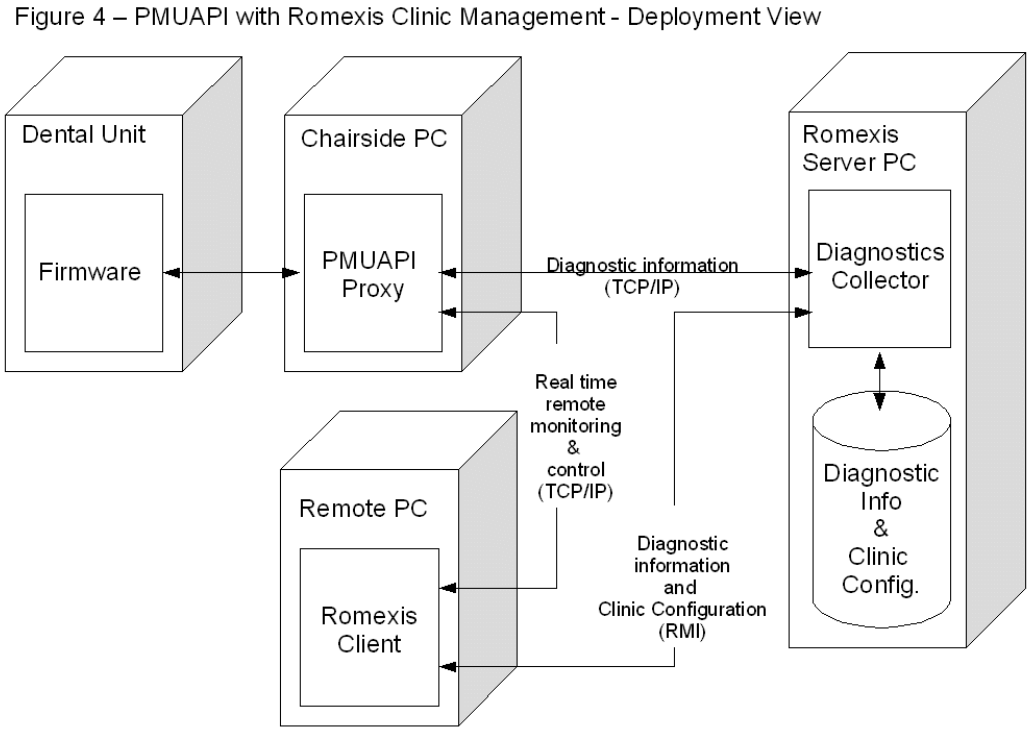
Kuva 8: Sovereignin web-käyttöliittymän instrumenttiasetusten muokkaussivu.

MultiProxy-nimellä, jos niitä on käynnissä useita samassa prosessissa. Ohjelmistoja voidaan käyttää erilaisissa kokoonpanoissa, joita on esitelty kuvissa 9 ja 10. Kuvissa hoitokoneena on Compact, joten siinä on mukana Planlink-ohjelmat, joista omat prosessinsa on palvelimella ja asiakasohjelmalla. Asiakasohjelman käyttämä Planlink voi, kuten kuvista näkyy, toimia eri koneella kuin itse asiakasaohjelma. Sovereignin kanssa Planlinkiä ei käytetä, vaan hoitokoneeseen otetaan suoraan yhteydet palvelimelta tai asiakasohjlemsta. Tällöinkin asiakasohjelma ja palvelin toimivat fyysisesti eri koneilla, mikä ei kuitenkaan ole välttämätöntä.

Romexis-ohjelmisto on toteutettu Java-kielellä käyttäen hyväksi sen natiiveja käyttöliittymäkirjastoja, Swingiä ja AWT:tä. Erilaiset painallukset ja muut vuorovaikutukset käyttöliittymän kanssa välittyvät eteenpäin ohjelmassa tapahtumina, jotka sitten ohjelma käsittelee. Tämän kyön kannalta oleellisimman Romexiksen osan eli asiakasohjelman Clinic Management -moduulin luokkarakennetta on havainnollistettu kuvassa 11. Tämän työn toiminnallisuus on sijoitettu `RoomPanel` ja `MonitoringPanel` välilehdille. Yhteyttä palvelimelle on myös pidettävä, koska sen yhteydessä sijaitsee tietokanta ja tiedostojärjestelmä, johon asetukset on tallennettu. Data asiakasohjelman ja palvelimen välillä siirtyy automaattisesti niiden välille pystytetyn RMI-rajapinnan kautta. [48] [6] [7] [5]

3.5 Sovereignin ja Romexiksen välinen viestirajapinta

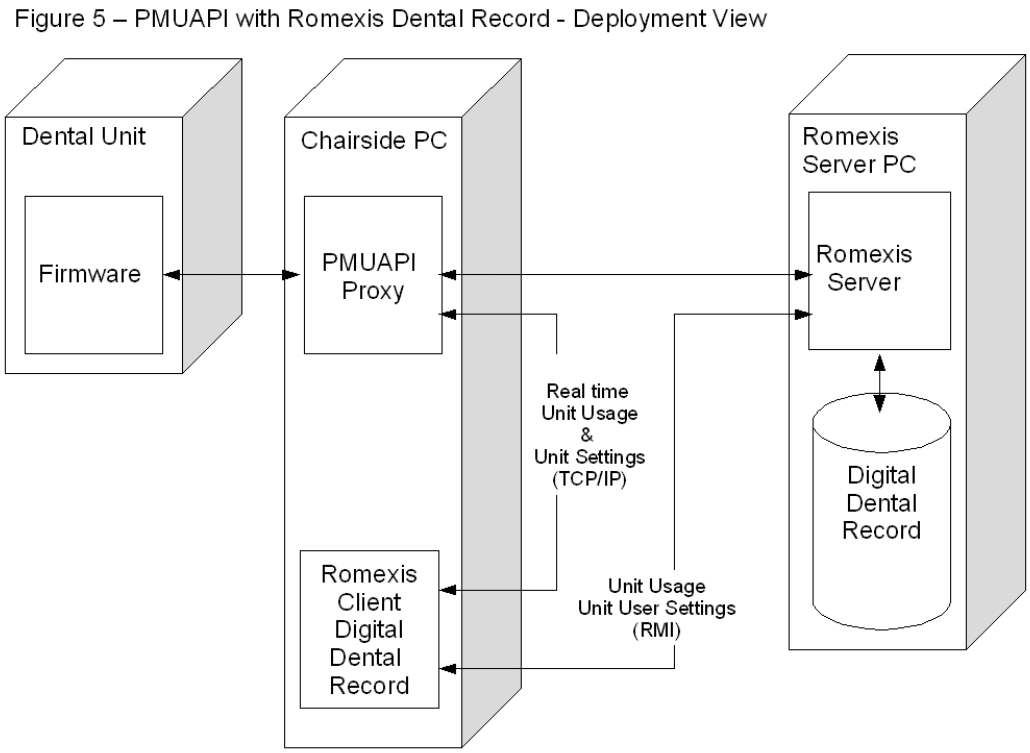
Sovereignin ja Romexiksen välinen keskustelu perustuu TCP-protokollaan. Sovereignin käynnistyessä se ottaa yhteyttä Romexis-palvelimeen. Tämän jälkeen palvelin pyytää hoitokoneelta lisätietoja sen kokoonpanosta ja asetuksista. Romexis-



Kuva 9: Esimerkki Romexis-ohjelmiston kokoonpanosta tapauksessa, jossa Planlink ja asiakasohjelma ovat eri tietokoneilla. [48]

asiakasohjelma saa palvelimelta nämä tiedot, jolloin asiakasohjelma voi aina tarvittaessa avata oman yhteydensä hoitokoneeseen. Sovereignin ja Romexiksen väliset viestit ovat pääosin tekstimuotoisia. Rajapintaan on määritetty tiettyjä määrämuotoisia viestejä, jotka on kummankin osapuolen tiedossa. Viestiliikennettä voi myös seurata avaamalla yhteyden oikeaan porttiin hoitokoneella. Porttiin voi myös kirjoittaa omia viestejään emuloiden näin Romexista. Sovereign hyväksyy kaikki yhteytensä Romexikseen Planlink-prosessissaan. Samassa prosessissa se myös käsittelee Romexikselta saamansa viestit.

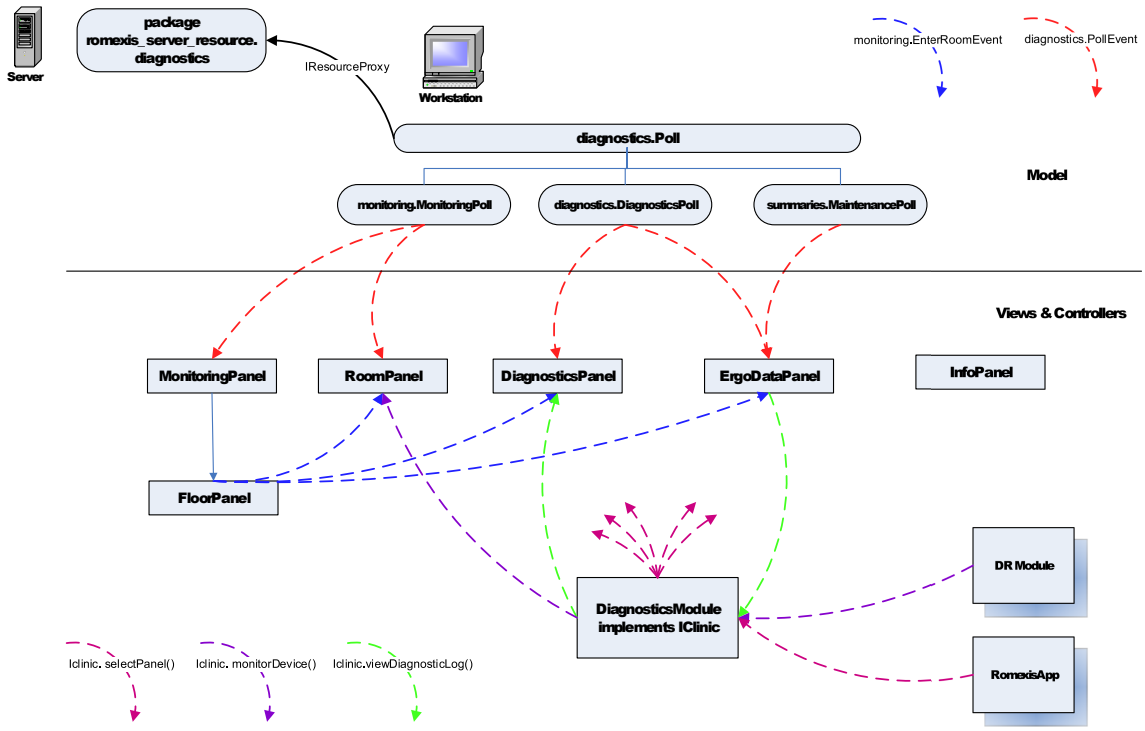
Tässä työssä Romexis esimerkiksi kysyy hoitokoneelta, voidaanko asetuksia siirtää lähettämällä viestin `req.setting.transfer`, johon hoitokone vastaa lähettämällä myöntävän vastauksen `rep.setting.transfer.ok` tai kielteisen vastauksen `rep.setting.transfer.nok`. Viestit saattavat sisältää myös välilyönneillä erotettuja avain-arvo-pareja, joissa avain ja arvo on erotettu toisistaan yhtäsuuruusmerkillä. Näillä voidaan välittää erilaisia lisätietoja viestien mukana. Esimerkiksi, kun Romexis ilmoittaa hoitokoneelle, että uusi asetustiedosto on lähetetty, se kertoo samalla asetustiedoston nimen ja sen, voidaanko kirjautumisesto purkaa ja tuleeko hoitokone uudelleenkäynnistää uusien asetusten käyttöönoton jälkeen. [49]



Kuva 10: Esimerkki Romexis-ohjelmiston kokoonpanosta tapauksessa, jossa Planlink ja asiakasohjelma ovat samalla tietokoneella. [48]

3.6 Haastattelut

Tässä työssä Sovereign-hoitokoneelle toteutettu ominaisuus oli siis jo aiemmin toteutettu ja käytössä Compact-hoitokoneelle. Ominaisuus ei kuitenkaan ollut vielä kovin laajassa käytössä. Haastateltaviksi ei yrityksestä huolimatta saatu lääkäreitä, huoltomiehiä, teknikoita, asentajia tai muita hoitokoneiden kanssa toimivia yrityksen ulkopuolisia henkilöitä. Sen sijaan aiheesta keskusteltiin yrityksen tuotekehitys- ja after sales -joukkojen kanssa. Ilmeisesti työssä tehdylle ominaisuudelle on kuitenkin ollut kysyntää ja sitä on toivottu. Ominaisuudesta voi olla hyötyä monella tapaa. Sen lisäksi, että sitä voivat lääkärit käyttää työssään isoilla klinikoilla, siitä on apua myös hammaslääketieteen oppilaitoksissa isoissa opetustiloissa, joissa opettajat voivat helposti asettaa kaikille oppilaille samat asetukset kaikkiin hoitokoneisiin yhdellä kertaa. Myös hoitokoneita huoltavat ja ylläpitävät teknikot hyötyvät ominaisuudesta sen helpottaessa heidän työtään. Yrityksen sisällä kokemusta on kuitenkin lähinnä asennustöistä, joissa ominaisuudesta on ollut huomattavaa apua, kun koneiden alkuasetukset ovat hoituneet kätevästi. Ennen ominaisuutta asetukset on jouduttu siis asettamaan joko käsin, USB-asetusominaisuutta hyväksikäyttämällä tai asennusten yhteydessä automaattisella asennusskriptillä. Keskusteluissa pohdittuja kysymyksiä on listattu alla. Luonnollisesti suurimpaan osaan kysymyksistä voidaan vastata vain Compactin osalta. Moniin kysymyksiin ei vielä saatu kokemuksen kautta vastauksia.



Kuva 11: Hahmotelma Romexiksen Clinic Management -moduulin luokkarakenteesta. [48]

- Miten ennen ominaisuuden toteutusta hoitokoneiden asetusten päivitykset on hoidettu?
- Tuliko idea ominaisuudelle asiakkailta vai keneltä?
- Minkälaisessa käytössä ominaisuus on ollut esimerkiksi yliopistoilla tai isoilla klinikoilla?
- Kuka ominaisuutta käyttää?
- Miten ja millaisissa tilanteissa sitä käytetään?
- Onko Compactin ja Sovereignin käytöissä jotain eroa?
- Mitä merkitystä ja hyötyä ominaisuudesta on?
- Mitkä ovat ominaisuuden vaatimuksest?
- Onko ominaisuudesta saatu jonkinlaista palautetta?

3.7 Asetustiedostot

Sovereign-hoitokoneen asetukset jaetaan kahteen päätyyppiin. Hoitokonekohtaiset asetukset ovat nimensä mukaisesti hoitokoneen asetuksia, jotka eivät vaihdu käyttäjältä toiselle. Nämä ovat siis esimerkiksi hoitokoneen käyttöliittymään tehtyjä muu-

toksia, jotka ovat samat kaikille yhden hoitokoneen käyttäjille. Nämä asetukset vaihtuvat koneelta toiselle, vaikka sama käyttäjä käyttäisi eri koneita. Käyttäjäkohtaiset asetukset ovat taas koneelle tallennettuja eri käyttäjien omia asetuksia. Varsinaisia käyttäjiä koneella voi olla kerrallaan viisi kappaletta. Lisäksi niin sanottua guest-käyttäjää käytetään, kun konetta käytetään ulkoiselle USB-Flash-muistille tallennetuilla asetuksilla. Asetukset on talletettu hoitokoneen Unix-pohjaisen käyttöjärjestelmän tiedostojärjestelmään tekstipohjaisiin tiedostoihin.

Kaikki asetukset löytyvät tiedostojärjestelmän yhdestä kansiosta ja sen alikansioista. Konekohtaiset asetukset löytyvät kaikki omasta alikansiostaan. Asetuksia löytyy vielä useasta eri tiedostosta, vaikka tarkoitus on vähentää tiedostojen määrää ja mahdollisesti saada kaikki asetukset sisällytetyksi yhteen tiedostoon. Tiedostoja on kahdeksan erillistä. Kaikki tiedostot siirretään yksinkertaisuuden vuoksi Romexis-palvelimelta Romexis-asiakasohjelman kautta hoitokoneelle ja takaisin. Kuitenkin asetuksia hoitokoneelle lähetettäessä vain osa tiedostoista lopulta korvataan ja yhdestä tiedostosta vain osa riveistä korvataan uusilla. Tämä johtuu siitä, että monet tiedostoista sisältävät asetuksia, jotka koskevat vain tiettyä hoitokonetta tai joiden ei muusta syystä haluta siirtyvän koneelta toiselle. Yksi tiedosto sisältää tietoja koneelle suoritetuista pesuista. Sitä ei siis korvata. Sen sijaan tiedostot, jotka sisältävät instrumenttien oletusasetukset, tuolin käsikäyttöisen ajon parametrit, tuolin eri moottoreiden käyttörajoitukset ja valokovettimen ajastusasetukset korvataan sellaisinaan. Tiedostoja, jotka kertovat vesilinjojen pesun puhdistusaineen vaikutusajan ja tiedon eri pesuihin liittyvistä tiloista ja edellisistä suoritetuista pesuista, ei myöskään haluta korvata, sillä nekin liittyvät selkeästi vain yhteen tiettyyn hoitokoneeseen.

Tärkein hoitokoneen asetustiedostoista sitten sisältää suurimman osan kaikista asetuksista. Tiedostosta poimitaan vain osa riveistä, joilla ylikirjoitetaan olemassaolevat valmiit rivit tai rivin puuttuessa lisätään uusi rivi. Kuten luonnollista on, siirrettäviä asetuksia ovat sellaiset, jotka eivät suoraan kytkeydy johonkin tiettyyn hoitokoneeseen. Tällaisia ovat muun muassa lääkärin tekemät muutokset hoitokoneen käyttöliittymään ja jotkin pienet muutokset laitteiston ja sen huoltotoimenpiteiden asetuksiin. Sen sijaan kaikki laitteiston kokoonpanoon ja sen kalibrointiin liittyvät asetukset jätetään kopioimatta. Myöskään hoitokoneen verkkoasetuksiin ei kosketa.

Käyttäjäkohtaiset asetukset löytyvät asetuskansion juuresta sekä sen alta löytyvistä käyttäjäkohtaisista hakemistoista, joita on omansa viidelle koneeseen tallennetulle käyttäjälle ja yhdelle USB-portin kautta asetuksensa tuovalle vierailevalle käyttäjälle. Kaikkien kansioiden sisältämät tiedostot ovat pääpiirteissään samat. Joitain eroja löytyy muun muassa sen mukaan, onko käyttäjän nimeä vaihdettu, onko käyttäjä muokannut eri instrumenttien tehdasasetuksia tai onko käyttäjällä jotain muutoksia tallentamatta. Käyttäjien kansioista löytyy iso lista tiedostoja, jotka liittyvät lähinnä instrumenttien asetuksiin ja esisäädettyihin tuolin asentoihin erikokoisia potilaita ja eri toimenpiteitä varten. Lisäksi asetuskansion juuresta löytyy tiedosto, joka sisältää kaikkien mahdollisten instrumenttien tehdas- eli oletusasetukset. Sitä ei siirretä asetuksia kopioitaessa. Sen sijaan kaikki eri käyttäjien omat kansiot kopioidaan ja asetuksia lähetettäessä korvataan suoraan. Jos yksit-

täinen käyttäjä on tehnyt muutoksia jonkin instrumentin oletusasetuksiin, löytyy käyttäjän kansiosta tiedosto, johon kaikki muutokset suhteessa oletusasetuksiin on listattu. Lisäksi, kun käyttäjä tallentaa muutoksensa, luodaan kansioon näille vielä oma tiedostonsa. Tämä on tarpeen, että instrumenttien käyttö toimisi halutulla tavalla. Tämän takia asetukset on myös oltava kullekin instrumentille erikseen kullekin mahdolliselle hoitokoneen instrumenttipaikalle. Käytössähän voi olla useita samantyyppisiä instrumentteja. Tallentamattomien instrumenttiasetusten halutaan säilyvän tietyissä käyttötapauksissa ja poistuvan toisissa. Esimerkiksi jos käyttäjä muuttaa jotain jonkin instrumentin asetuksista, käyttää sitten toista instrumenttia ja palaa ensimmäiseen instrumenttiin, ovat tehdyt muutokset edelleen voimassa. Ne ovat säilyneet tiedostossa, jossa on tallentamattomat käyttäjän tekemät muutokset. Toisaalta jos käyttäjällä on tallentamattomia muutoksia jossakin esiasetuksessa ja valitsee sitten toisen esiasetuksen, ensimmäisen esiasetuksen tallentamattomat muutokset menetetään.

Käyttäjäkohtaisista asetuksista löytyy myös tiedosto, joka sisältää käyttäjänimen, jos käyttäjä on tallentanut sellaisen itselleen. Yhdestä tiedostosta löytyy tieto siitä, minkä kätiselle lääkärille hoitokone on asennettu ja säädetty. Loput asetustiedostoista liittyy hoitokoneen tuolin asentoihin ja ajoihin. Tiedostot sisältävät esitallennettuja tuolin asentoja erikokoisille potilaille ja erilaisille toiminnoille, kuten ylätai alaleuan tutkimiselle tai röntgen-kuvien ottamiselle. [50] [51]

Hoitokoneen on myös suunniteltu tukevan eri niin sanottuja hoitotiloja, kuten kirurginen tila kirurgisia toimenpiteitä varten ja kuvaustila hampaiden ja leuan kuvantamista varten. Kussakin hoitotilassa voisi käytössä olla eri instrumentit ja eri asetukset niille. Lisäksi tuolin valmisasennot olisivat omansa. Hoitotilaa vaihdettaessa myös kaikki tallentamattomat muutokset menetettäisiin. Tällä hetkellä tosin hoitokone tukee vain yhtä yleistä hoitotilaa. Jotta eri hoitotilat voitaisiin toteuttaa, pitäisi luultavasti laittaa alikansioihin omat asetustiedostot kullekin hoitotilalle. [45]

Työtä tehdessä pohdittiin mahdollisuutta yksinkertaistaa hoitokoneen asetuslogiikkaa esimerkiksi vähentämällä tarvittavien tiedostojen määrää. Muun muassa tavoitteena oli mahduttaa konekohtaiset asetukset yhteen tiedostoon. Toisaalta olisi ollut kätevää, jos kalibraatioasetukset löytyisivät yhdestä tiedostosta, muut tiettyyn koneeseen sidotut asetukset toisesta ja kaikki siirrettäväksi tarkoitetut asetukset kolmannesta omasta tiedostostaan. Kaikki nämä muutokset tulisi samalla suhteuttaa yhtiön tuleviin hoitokoneisiin ja pyrkiä yhteensopivuuteen ja uudelleenkäytettävyyteen. Muutoksiin ei kuitenkaan ryhdytty tässä vaiheessa, koska ominaisuus haluttiin ensin toteuttaa toimivana mahdollisimman pian ja vasta sitten lähteä mahdollisesti kehittämään sitä eteenpäin. On myös mahdollista, että Sovereignin ohjelmisto korvataan uudella, jolloin vanhaan ohjelmistoon ei haluta kuluttaa turhaan työtunteja.

3.8 Ominaisuuden vaatimukset

Työn aloitusvaiheessa ei ollut aivan selvää, millainen asetustenhallintaominaisuus haluttiin toteuttaa, miten sen haluttiin toimivan ja millaisia ominaisuuksia siltä vaadittiin. Näitä asioita mietittiinkin eri työryhmissä useasti ja yritettiin selvittää toivomuksia eri henkilöiltä. Eräässä vaiheessa haluttiin, että ominaisuutta voitai-

siin käyttää joko pelkästään tai Romexiksen lisäksi myös Sovereignin käyttöliittymästä. Lääkärin tullessa koneelle hän olisi voinut hakea koneelle haluamansa asetukset Romexis-palvelimelta. Romexis olisi voinut lähettää Sovereignille kirjautuneelle käyttäjälle valittavana olevat asetukset ja lääkäri olisi voinut valita, mihin viidestä käyttäjäpaikasta asetukset olisi otettu käyttöön. Samoin lääkäri olisi koska vain voinut tallentaa ja lähettää asetukset Romexikselle. Lääkäri olisi syöttänyt asetuksille nimen, jonka perusteella hän olisi voinut myöhemmin tunnistaa tallennuksen. Tämä olisi vaatinut mahdollisesti Romexikseen tarkistuksen, että annettu nimi oli uniikki. Konekohtaisia asetuksia ei olisi lähetetty. Ominaisuus olisi vaatinut runsaasti muutoksia myös Sovereignin käyttöliittymään ja Romexikseen esimerkiksi tietokantojen osalta. Kaikki tallennetut asetukset olisi pitänyt pystyä linkittämään uniikisti johonkin hoitokoneen käyttäjään ja itse hoitokoneeseen. Tätä toteutettaessa olisi todennäköisesti jouduttu mahdollisesti kiertämään Romexis-asiakasohjelma, ja laittaa hoitokone keskustelemaan suoraan Romexis-palvelimen kanssa. Romexis-asiakasohjelmahan linkkaa tallennetut asetukset vain Romexikseen kirjautuneena olevaan käyttäjään. Ominaisuudet päätettiinkin jättää toteuttamatta ainakin tässä vaiheessa.

Lopulta siis päädyttiin keskittymään ominaisuuden toteuttamiseen käytettäväksi Romexiksen päästä. Mallina oli siis valmis toteutus Compact-hoitokoneelle, johon mietittiin parannuksia. Mietittiin muun muassa sitä, kuinka helposti erotettaviksi halutaan tehdä konekohtaisten ja käyttäjäkohtaisten asetusten haku ja palautus, ja halutaanko yhden koneen eri käyttäjien asetukset tehdä yksittäin palautettaviksi vai pitääkö kaikki käyttäjäasetukset palauttaa yhdellä kertaa. Harkitiin myös tallennettujen asetusten linkkaamista jollain tavalla tiettyyn koneeseen ja käyttäjäasetusten käyttäjiin. Kun hoitokoneen ohjelmiston versionumero tallennetaan kullekin haetulle asetukselle, voitaisiin tätä tietoa käyttää myös varmistamaan asetusten yhteensopivuus koneen version kanssa. Yhteensopivuksia ja kunkin version välisiä muutoksia ei ollut kunnolla tiedossa ja tämän ominaisuuden toteutus olisi vaatinut jonkinlaisen asetusten versionhallinnan rakentamisen ja runsaasti ylläpitotyötä tulevaisuudessa, joten tästäkin luovuttiin ainakin toistaiseksi. Samaan aiheeseen liittyy myös idea siitä, että asetustiedostojen sisältöjä olisi jotenkin tarkistettu mahdollisten virheellisten arvojen tai puuttuvien tietojen varalta. Samalla olisi tunnistettu kokonaan vääränmuotoiset tiedostot. Ominaisuus oltaisiin voitu tehdä joko Romexiksen tai Sovereignin päähän. Luonnollisempi lienee Sovereign, koska siellä on helpompi olla selvillä, mitä asetuksia juuri sen ohjelmistoversio ja kokoonpano tarvitsee. Tätäkään ideaa ei tässä vaiheessa vielä toteutettu, mutta konekohtaisten asetusten kohdalla Sovereign osaa lisätä puuttuvat asetusarvot tietyillä koodista löytyvillä oletusasetuksilla. Tämä olisi varmasti hyödyllinen toiminto myös käyttäjäkohtaisten asetusten osalta.

Eräs mietitty parannus oli se, että sen lisäksi, että hoitokoneessa voidaan käyttää käyttäjäasetuksia USB-muistitikulta, näin voisi tehdä myös Romexis-koneen kautta. Eli USB-tikun asetukset olisi voinut ottaa talteen Romexikselle tai lähettää suoraan hoitokoneelle, samalla hoitokoneelta olisi voitu tallentaa erillisenä USB-muistitikulla olleet asetukset. Nämä parannukset olisivat vaatineet suurempia muutoksia Romexiksen käyttöliittymään ja sen taustalla olevaan logiikkaan ja tietokan-

toihin. Tällöin ominaisuus olisi myös käyttäjän näkökulmasta toiminut eri tavalla eri hoitokonetyypeille. Tähän ei siis lähdetty, vaan lopulta ominaisuus päätettiin toteuttaa ulkoisesti samalla tavalla kuin Compactin vastaava ominaisuus. Näin päästiin jälleen vähemmällä työllä ja pienemmillä muutoksilla olemassa olleisiin rakenteisiin. Pinnan alla toteutus oli toki erilainen Compactiin verrattuna. Ominaisuuden toteutuksesta ja sen eroista kerrotaan lisää luvuissa 4 ja 5. Vaikka ominaisuuden haluttiinkin toimivan samalla tavalla kuin Compactin kohdalla, mietittiin kyllä kuitenkin Romexiksen käyttöliittymään yleisempiä parannuksia, joista kerrotaan myös lisää myöhemmin. Muutoksiin lähdetään ehkä tulevaisuudessa suuremman Romexiksen käyttöliittymäremontin yhteydessä.

Kun ominaisuuden vaatimukset oli määritelty, jouduttiin vielä miettimään joitain käytännön kysymyksiä ominaisuuden käyttöön liittyen. Ensimmäinen kysymys oli se että, miten suhtaudutaan hoitokoneen käyttämiseen asetuksia haettaessa tai päivitettäessä. Saisiko asetuksia hakea tai ennen kaikkea, olisiko turvallista lähettää asetuksia, kun hoitokone oli käytössä? Olisi ollut mahdollista kysyä käyttäjältä, voitaisiinko asetukset siirtää tai haluttiinko uudet asetukset ottaa käyttöön. Turvallisuuden varmistamiseksi ja jälleen myös yksinkertaisuuden vuoksi päätettiin, että hoitokoneella ei saisi olla asetuksia siirrettäessä kumpaankaan suuntaan käyttäjää kirjautuneena. Lisäksi päätettiin, että ei mahdollisteta käyttäjän uloskirjaamisen pakottamista Romexiksesta käsin, mistä kieltämättä olisi huomattavasti apua, jos voidaan esimerkiksi olla varmoja, että suuren opetustilan millään koneella ei ole kukaan toimimassa, vaikka kaikissa olisikin käyttäjä kirjautuneena. Nyt joudutaan siis käydä käsin kirjaamassa jokainen käyttäjä erikseen ulos. Hoitokone lukitaan asetustensiirtoa aloitettaessa siten, että koneelle ei voi kirjautua kesken prosessin. Jos prosessi jostain syystä keskeytyy tai jokin viesti ei mene perille asti, saattaa käydä niin, että hoitokone jää tähän lukitustilaan ja vaatii uudelleenkäynnistyksen koneen käyttämiseksi. Yksi tapa estää tämä, olisi liittää asetustensiirtoon joku aikaraja, jonka jälkeen lukitus avautuisi automaattisesti joka tapauksessa. Tällöin pitäisi pitää myös huoli, että asetustensiirtoa ei jatketa, jos viesti vielä tulee vaan hoitokoneen tila palautuisi siirron aloittamista edeltävään tilaan. Asetuksien käyttöönottoon liittyy myös hoitokoneen uudelleenkäynnistys. Käyttäjäkohtaiset asetukset tulevat käyttöön ilman uudelleenkäynnistystä, mutta konekohtaiset asetukset vasta käynnistyksen jälkeen. Päätettiinkin, että hoitokone käynnistää itse itsensä vain konekohtaisten asetusten lähettämisen jälkeen. Jos lähetetään samalla kertaa myös käyttäjäkohtaiset asetukset, konekohtaiset lähetetään jälkimmäisenä. Koneen käyttöliittymä ei varoita tulevasta uudelleenkäynnistyksestä eikä täten myöskään pyydä vahvistusta siihen, mikä helpottaa jälleen päivittäjien työtä isoja konemääriä kerralla päivitettäessä, eikä varoituksen puutteella väliä olekaan, koska koneella ei kirjautunutta käyttäjää ole, eikä kone siis ole käytössä. Toteutuksen suunnitteluun liittyy myös joitain asioita, joita mietittiin enemmänkin. Näistä kerrotaan lisää tämän luvun seuraavissa osissa. [52]

3.9 Tiedonsiirto

Vaikka monien eri työssä tehdyn ominaisuuden toteutettujen osien kohdalla tutkittiin ja harkittiin eri toteutustapoja ja -vaihtoehtoja, erityisesti käytettyjen tek-

nologioiden valinnan useimmissa tapauksissa saneli kuitenkin tuotteessa jo käytössä olevat teknologiat ja yrityksestä löytynyt valmis osaaminen tietyissä asioissa. Asetustiedostojen siirtämiseen hoitokoneen ja Romexiksen välillä harkittiin ensimmäisenä TFTP:n eli Trivial File Transfer Protocolin käyttöä. TFTP:n käyttöönottoa varten oli jo tehty valmisteluja yhteyden kumpaankin päähän, niin Sovereign ACCU-kortille kuin Romexikseenkin. Yhteys ei kuitenkaan ollut täysin valmis ja toimiva, joten hieman lisätyötä sen käyttöönottamiseksi olisi vaadittu varsinaisen ominaisuuden toteuttamisen lisäksi.

Lopulliseksi toteutustavaksi valikoitui melko helposti HTTP-protokollan käyttö. Sovereign-hoitokoneelle on käytössä selainpohjainen käyttöliittymä, josta voidaan muun muassa vaihtaa useita eri konekohtaisia asetuksia. Selain keskustelee luonnollisesti hoitokoneen kanssa HTTP:tä käyttäen ja hoitokoneella on tätä varten käynnissä web-palvelin. Käyttöliittymä kutsuu palvelimelta CGI-skriptejä, joiden käyttö oli siis jo valmiiksi mahdollistettu. Heti alusta oli selvää, että hoitokoneen asetustiedostojen vaihto haluttaisiin jossain vaiheessa tehdä mahdolliseksi myös selaimen kautta. Siksi oli luonnollista lähteä kehittämään myös Romexiksen osalta tiedostonsiirtoa HTTP:lla, jolloin ominaisuuden toteuttaminen tulevaisuudessa selaimelle onnistuisi vähemmällä työllä uudelleenkäyttäen olemassaolevia moduuleita. Myös CGI:tä päätettiin samasta syystä käyttää hoitokoneen päässä. Siitä löytyi siis jo yrityksestä osaamista ja kirjallisuutta. Selaimen käyttöliittymän kanssa käytetyt CGI-skriptit oli tehty Perl-kieltä käyttäen, mutta tässä työssä käytettiin C++ kieltä, koska sille löytyi sopiva CGICC-kirjastoa hyväksikäyttävä esimerkki, jonka pohjalle skripti oli helppo rakentaa. Kirjaston valmiilla funktioilla HTTP-vastauksen ja HTML:n luominen ja tiedoston ja HTML-lomakkeen käsittely oli helppoa. Kirjastosta löytyi yrityksestä myös kokemusta. Kirjasto on esitetty tarkemmin luvussa 2.6.2. C++-kieli oli työn tekijälle ennestään tuttu toisin kuin Perl. Lisäksi samoja skriptejä pystyttiin myös myöhemmin käyttämään hoitokoneen päivittämiseen.

Lisää vaihtoehtoja ja niiden ominaisuuksia tiedonsiirrolle on esitelty ja vertailtu luvussa 2.3. Näitä vaihtoehtoja mietittiin ja niistä otettiin selvää. Lähinnä vaihtoehdot olisivat hyötyinä tarjonneet parempaa tietoturvaa. Osa vaihtoehdoista olisi tarjonnut turhaan myös paljon tarvittua enemmän ominaisuuksia. Useimmat niistä olisivat myös olleet huomattavasti työläämpiä toteuttaa, ja ulkoisia kirjastoja olisi tarvittu tai pahimmassa tapauksessa protokolla olisi pitänyt itse toteuttaa alusta asti. Ominaisuus pyrittiin tekemään kuitenkin mahdollisimman vähällä työllä. Samalla pyrittiin yksinkertaisuuteen ja mahdollisimman pieniin muutoksiin olemassaolleisiin sovelluksiin. Näin pystytään yleensä minimoimaan virheet, saamaan toimiva ominaisuus, helpottamaan ominaisuuden testaamista ja maksimoimaan tehdyn koodin uudelleenkäytettävyys. Ominaisuus oli jo toteutettu Romexikseen yrityksen toiselle hoitokoneelle, Compactille, ja sen haluttiin toimivan mahdollisimman pitkälle käyttäjän näkökulmasta samoin tavoin. Tämä tietenkin helpottaa ja parantaa käyttäjäkokemusta ja vähentää käyttöohjeiden tekijöiden työtä. Useimpien muiden tiedonsiirtomenetelmien käyttö olisi siis varmasti vaatinut myös paljon pohjatyötä tarvittavan alustan rakentamisessa teknologian käyttöönottoa varten. Olisi varmasti jouduttu muun muassa käyttämään lisää ulkoisia kolmannen osapuolen kirjastoja, mikä on aina erityisesti lääketieteellisissä laitteissa ongelmallista, kun ulkoisten kir-

jastojen pitää myös täyttää lääketieteellisten ohjelmistojen turvallisuusvaatimukset. Työ olisi lisääntynyt ainakin dokumentaatiossa ja testauksessa, ja vikojen mahdollisuus olisi kasvanut. On myös hyvin mahdollista, että monien vaihtoehtojen kohdalla, ei olisi ollut olemassa valmiita toteutuksia käytössä olleille alustoille ja ohjelmointikielille. Lisätyö olisi ollut erityisen kallista, koska siitä ei olisi todennäköisesti ollut hyötyä kovin monessa eri tilanteessa. Yksi tälläinen olisi ollut hoitokoneen päivitystiedoston lähettäminen etänä, mutta nyt sekin siis hoidetaan HTTP:n avulla. Selkeä etu monissa muissa vaihtoehdoissa olisi ollut valmis yhteyden salaus, joka on toki mahdollista toteuttaa myös HTTP:tä käyttäen.

CGI-skriptejä toteutettaessa päädyttiin miettimään, voisiko skripti keskustella itse suoraan Sovereignin pääohjelmiston kanssa ilmoittaen saapuneista asetuksista tai kysyen, onko koneelle kirjautuneena ketään. Tämä olisi lisännyt ominaisuuden monimutkaisuutta sekä vaadittua työtä. Skriptien ja Sovereignin välille olisi pitänyt kehittää uusi rajapinta. Romexis laitettiin siis keskustelemaan jo olemassaolleen yhteyden kautta suoraan Sovereignin kanssa. Tulevaisuudessa tosin, jos halutaan mahdollistaa asetusten päivitys selaimen kautta, on skriptien ja Sovereignin välinen yhteys toteutettava.

3.10 Asetustietojen tallennustapa

Asetustietojen tallennustapaa pohdittiin myös pitkään. Aluksi mietittiin, olisiko asetuksissa esiintyville avain-arvo-pareille voitu tehdä kullekin omat kentät yhteisesssä tietokannassa. Tämä olisi vaatinut runsaasti ylläpitoa kannan päivityksen muodossa, kun käytössä olevat asetukset olisivat saattaneet vaihtua versioiden välillä. Olisi saattanut tulla myös tarpeelliseksi pitää yllä useampia eri tauluja eri versioille. Seuraava vaihtoehto oli sitten oma taulu tai Compactin asetustauluun oma kenttä, johon asetustiedostojen sisällöt olisi tallennettu. Muutoksia tietokantoihin ei kuitenkaan haluttu tehdä, joten oli tyydyttävä olemassaolevaan kantaan. Tässä vaiheessa olisi ollut mahdollista rakentaa jonkinlainen järjestelmä, jossa tietokantaa ei olisi käytetty lainkaan, vaan pelkästään tiedostoja. Compactin asetuksia varten oli jo olemassa taulut tietokannassa. Tauluja on kaksi, joista ensimmäiseen tallennetaan yksi merkintä per käyttäjä- tai konekohtainen asetus ja toiseen useampia rivejä lisätietoja kustakin asetuksesta ja Compactilla itse asetukset. Compactilla tietokannan tauluihin tallennettiin siis hoitokoneelta suoraan bittivirtana saadut asetukset. Compact lähettää niitä Romexikselle tasaisin väliajoin. Bitit koodataan tavuittain tekstiksi ja tallennetaan taulun yhteen kenttään. Compactin käyttämässä taulussa oli siis kyllä kenttä, johon myös Sovereignin dataa olisi voitu tallentaa. Sen maksimikoko oli kuitenkin 2000 merkkiä. Pidempi data jaetaan kyllä automaattisesti useammalle riville, mutta tiedoston sisältö olisi pitänyt jakaa niin monelle riville, että päädyttiin tallentamaan kantaan pelkkä polku asetustiedostoon, kun itse tiedosto tallennettiin Romexis-palvelimen tietokoneen käyttäjän kotikansioon.

Toisaalta oli pohdittava, mitä hoitokoneen ja Romexiksen välillä siirretään. Oliko tarkoitus siirtää yksittäisiä tiedostoja, yhteen pakattuja tiedostoja vai yksittäisiä asetusten arvoja. Päädyttiin siirtämään erillisissä pakkauksissa yhdessä kaikki konekohtaiset ja kaikki käyttäjäkohtaiset asetukset. Mietittiin myös, millä tekniikalla

tiedostot pakattaisiin yhteen. Sovereignin Unix-pohjaisessa käyttöjärjestelmässä tar-tiedoston luominen oli skriptillä helppoa, mutta sen purkaminen Romexiksen Java-koodissa oli hankalampaa. Se olisi vaatinut ulkoista kirjastoa, jos sellaista olisi edes ollut saatavilla. Valitussa ratkaisussa ei kuitenkaan ollut tarpeen tiedoston purkaminen, joten tar-teknologian käyttäminen oli lopullinen vaihtoehto esimerkiksi zip- tai gzip-pakkauksen sijaan. Yksittäisen tiedostopaketin siirtäminen helpottaa myös selainkäyttöliittymän valmistelemista asetustensirtoa varten. Toisaalta tar-pakkaus ei itse asiassa lainkaan pakkaa tiedostoa pienemmäksi vaan kasaa vain tiedostot yhteen. Tiedostokoko ei ole kuitenkaan osoittautunut tässä tapauksessa ongelmaksi.

Ratkaistavana oli myös se, miten tietoa siirrettiin Romexis-asiakasohjelman koneelta Romexis-palvelimen koneelle. Eri vaihtoehtoja ei tarvinnut kuitenkaan tarkemmin selvittää, sillä nopeasti selvisi, että siirtämiseen voitiin käyttää jo olemassaollutta RMI-rajapintaa, joka hoiti automaattisesti olioiden siirron ja metodien kutsun erillisten koneiden välillä. Asetustiedostojen sisältö voitiin siis paketoida olioihin, jotka siirtyivät metodikutsujen mukana rajapinnan läpi. Myös alun epäilys, että näin suurten tietomäärien siirto hidastaisi sovellusta, osoittautui vääräksi.

3.11 Tietoturva ja potilasturvallisuus

Työssä käytetyissä toteutuksissa ja teknologioissa on monissa tietoturvan kannalta parantamisen varaa. Suurin osa niistä oli jo valmiiksi käytössä, eikä niitä haluttu lähteä vaihtamaan. Kuitenkin niiden heikkoudet tunnistettiin. Monilla tietoturva-aukoilla on potentiaalisia väärinkäytön mahdollisuuksia, joten ne voivat olla myös mahdollinen potilasturvallisuusriski, jos esimerkiksi hoitokoneen asetuksia päästään sorkkimaan tai sen ohjelmistoa muuttamaan tai tuhoamaan. Toki monet näistä asioista tekee tuotteen kehityksestä ja testauksesta helpompaa, mutta ehkäpä olisi mahdollista pitää kehitys- ja tuotantoversiot näiltä osin erilaisina.

Ensinnäkin, kun Romexis lähettää Sovereignin HTTP-palvelimelle pyynnön lähettäen samalla kirjautumistiedot, käytetäään `basic`-tyyppistä kirjautumista. Tällöin käyttäjätunnus ja salasana lähetetään selkokielinen teksti vain kevyesti koodattuna BASE64-algoritmilla. Kirjautumistietoja ei ole siis mitenkään salattu tai sekoitettu eli hashätty ja lisäksi BASE64-koodaus on nykyaikaisilla tietokoneilla helppo purkaa. Olisi siis siirryttävä käyttämään `digest`-tyyppistä kirjautumista, jossa salasana on sekoitettu. Vielä parempi olisi vaihtaa lisäksi käyttämään SSL-suojattua HTTPS-yhteyttä. Myös säilytettävät salasanat voisi suolata eli lisäämällä niihin ennaltamäärätty osa, jolloin niiden selvillesaaminen vaikeutuisi entisestään. Tällä hetkellähän tunnukset löytyvät Java-koodin seasta, josta ne voidaan luultavasti kaivaa esille. [53] [54] [55] [56] [9]

Toisaalta hoitokoneeseen voidaan ottaa helposti yhteys Telnetillä, jolloin sen käyttö- ja tiedostojärjestelmään päästään käsiksi ja vaaratilanteiden luominen tulee mahdolliseksi. Telnet-yhteyden kautta voidaan myös lähettää hoitokoneelle Planlink-rajapinnan mukaisia viestejä, jolloin voidaan tekeytyä Romexikseksi ja tulevaisuudessa mahdollisesti vaikkapa ohjata hoitokonetta etäältä. Yhteys olisi vähintään vaihdettava huomattavasti turvallisempaan SSH-yhteyteen.

Itse Romexiksen päässäkin voidaan tehdä tuhoa. Palvelimelle tallennettujen ase-

tustiedostojen sisältöä voidaan helposti muuttaa. Tar-pakkaukset voisi suojata salasanalla. Tällöin kuitenkin olisi edelleen mahdollista yksinkertaisesti korvata tiedostot. Voisi mahdollisesti pyrkiä kuitenkin saamaan myös tiedostojen sisältö tietokantaan turvaan. Tietokantakaan ei sinällään ole äärimmäisen turvallinen sillä sen tunnukset löytyvät selkokielisinä Romexiksen asetustiedostoista.

Näillä heikkouksilla ja puutteilla on siis selkeästi olemassa mahdollisuus ainakin haitantekoon, jos ei kuitenkaan oikeiden vaaratilanteiden luomiseen. Sovereignin ja Romexiksen välillä ei ainakaan vielä siirretä luottamuksellisia tai muitakaan potilastietoja, joten potilaan tietoturva ei ainakaan ole näillä näkymin vaarassa. Ehkäpä tulevaisuudessa asia on kuitenkin pidettävä mielessä, vaikkakin hoitokoneita käytetään lähinnä suljetuissa lähiverkoissa. [57] [58] [9]

4 Ominaisuuden toteuttaminen

4.1 CGI-skriptit

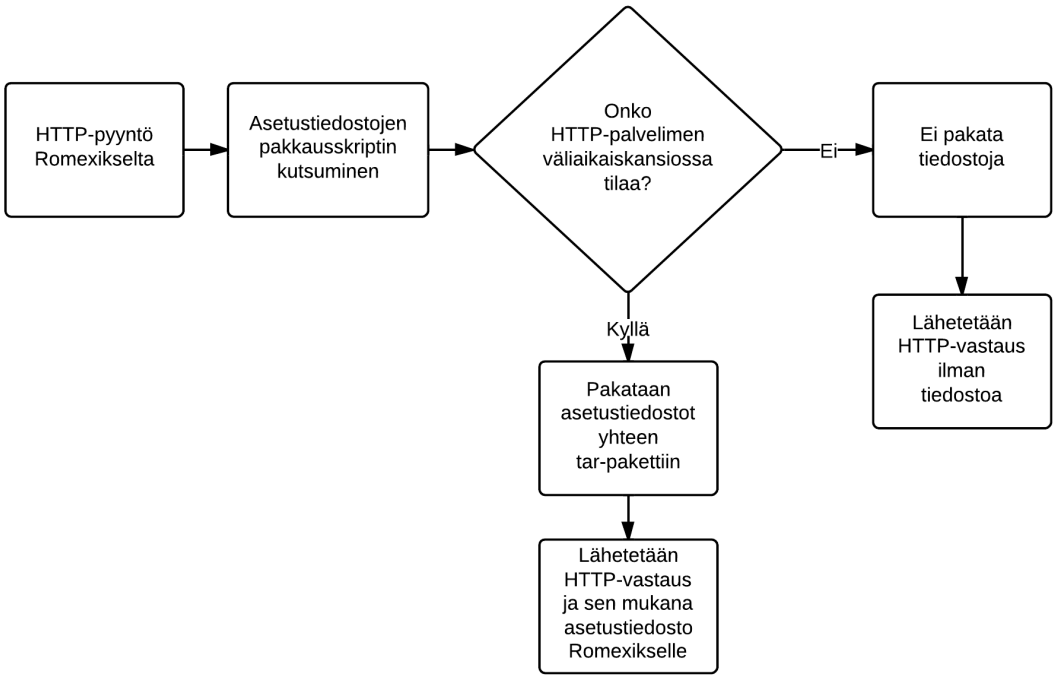
Ominaisuuden toteuttaminen aloitettiin CGI-skriptien kirjoittamisella. Skriptit kirjoitettiin C++-kielellä ja avuksi otettiin CGICC-kirjasto. Ensiksi kirjoitettiin skriptit konekohtaisten asetusten lähettämiseen Sovereignilta ja vastaanottamiseen Sovereignille. Käyttäjäkohtaisten asetusten CGI-skriptit saatiin näistä hyvin pienin muutoksin. Ainoa ero on asetustiedostoja yhteen tar-tiedostoon pakattaessa käytettävä shell-skripti. Toinen skripti pakkaa konekohtaisten asetusten kansion sisällön tar-tiedostoon, jonka nimi sisältää tiedon hoitokoneen nimestä, asetusten tyypistä ja niiden hakuajasta. Lisäksi tiedostonimeen lisätään satunnainen luku, jotta vältettäisiin mahdolliset samannimiset tiedostot. Tiedosto laitetaan väliaikaiseen säilytykseen HTTP-palvelimen tilapäiskansioon. Käyttäjäkohtaisten asetusten tar-tiedostoon pakataan vastaavasti toisella skriptillä kaikki käyttäjäkohtaisia asetuksia sisältävät asetuskansion alikansiot.

Asetustensiirtoon käytettyjen CGI-skriptien toiminta on esitetty yleisellä tasolla kuvissa 12 ja 13. Sekä konekohtaisten asetusten että käyttäjäkohtaisten asetusten hakuskripteissä tar-tiedoston luomisen eli pakkausskriptin kutsumisen jälkeen rakennetaan Romexikselle lähetettäväksi HTTP-vastaus, joka sisältää rungossaan luodun tiedoston. Vastauksen mukana lähetetään myös tiedoston nimi, joka saadaan pakkausskriptin palauttamana arvona. Asetuksia hoitokoneelle lähetettäessä taas tar-tiedosto saapuu HTTP-pyynnön mukana, ja se tallennetaan palvelimen `tmp`-kansioon. Lisäksi asetukset vastaanottava skripti sisältää paljon koodia, joka luo selaimesta käytettävää käyttöliittymää varten HTML-sisältöä, jonka mukaan selain näyttää lomakkeen, jolla voidaan lähettää haluttu tiedosto hoitokoneelle. Kun nämä shell-skriptit ja C++:lla kirjoitetut CGI-skriptit olivat valmiit, oli vielä automatisoitava Sovereignin ACCU-kortin päivitysprosessi niin, että päivityspaketin rakentava skripti käänsi myös CGI-skriptit ristiinkääntäjällä ja lisäsi ne ja CGICC-kirjaston pakettiin mukaan.

4.2 Romexis ja Sovereign

Seuraavaksi rakennettiin toiminnallisuus Romexikseen. Käyttöliittymä oli jo valmis, koska ominaisuus oli jo tehty Compact-hoitokoneelle. Oli siis vain toteutettava itse logiikka käyttöliittymän takana. Tähän ei ollutkaan lainkaan apua Compactin toteutuksesta, sillä se toimi aivan eri tavalla. Myös kaikkein alimmainen toiminnallisuus asetusten tallentamisessa tietokantaan ja sieltä hakemisessa vaati lopulta hieman muutoksia ja lisäyksiä. Compactilla asetukset sisältävät bitit oli tallennettu suoraan kantaan, mutta Sovereignilla asetustiedostot siirretään siis asiakaskoneelta palvelimelle ja tallennetaan sille suoraan tiedostoina. Tietokantaan tallennetaan ainoastaan tiedoston absoluuttinen polku. Varsinainen asetusten lähetys- ja vastaanottologiikka muodostettiin `Sovereign`-luokkaan. Näiden lisäksi oli toteutettava ja päivitettävä asetuksiin liittyviä apuluokkia.

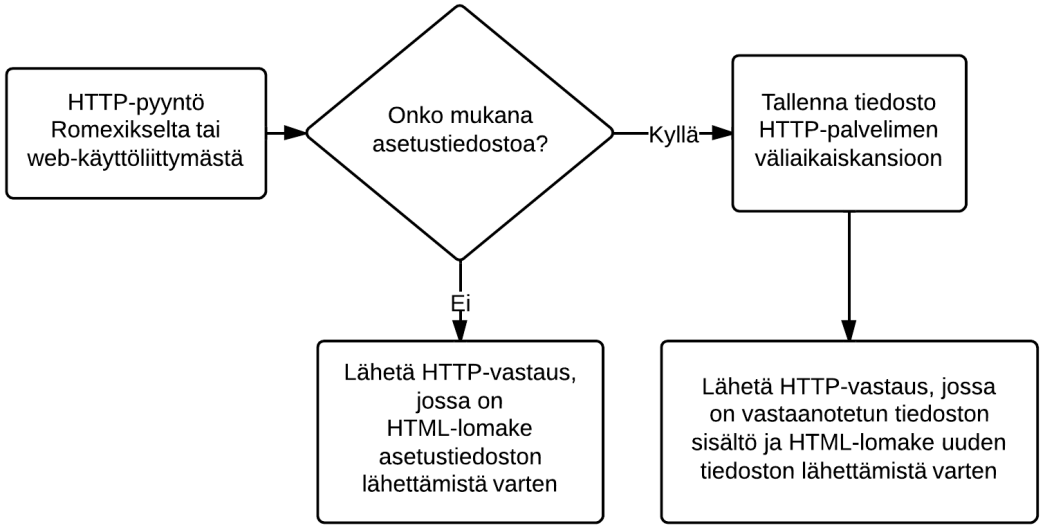
Romexiksesta löytyy asetuksiin liittyvää toiminnallisuutta kahdesta eri paikas-



Kuva 12: Asetustiedoston hakemiseen käytetyn CGI-skriptin toiminta. [59]

ta. `Unit Monitoring`-näkymästä voidaan yhdestä napista samalla kertaa hakea sekä valittuna olevan hoitokoneen konekohtaiset asetukset että käyttäjäkohtaiset asetukset. Toisesta napista lähetetään pelkät käyttäjäkohtaiset asetukset. Tämä on peruja Compactin käytöstä, sillä Compactilla ei konekohtaisia asetuksia pystytty lähettämään yhtä aikaa käyttäjäkohtaisten asetusten kanssa. Käyttöliittymän toiminnallisuus haluttiin kuitenkin pitää yhtenevänä hoitokoneiden välillä. `Clinic Monitoring`-näkymästä voidaan taas valita kerralla useampia hoitokoneita ja lähettää niille kaikille samat käyttäjä- ja konekohtaiset asetukset samalla kertaa.

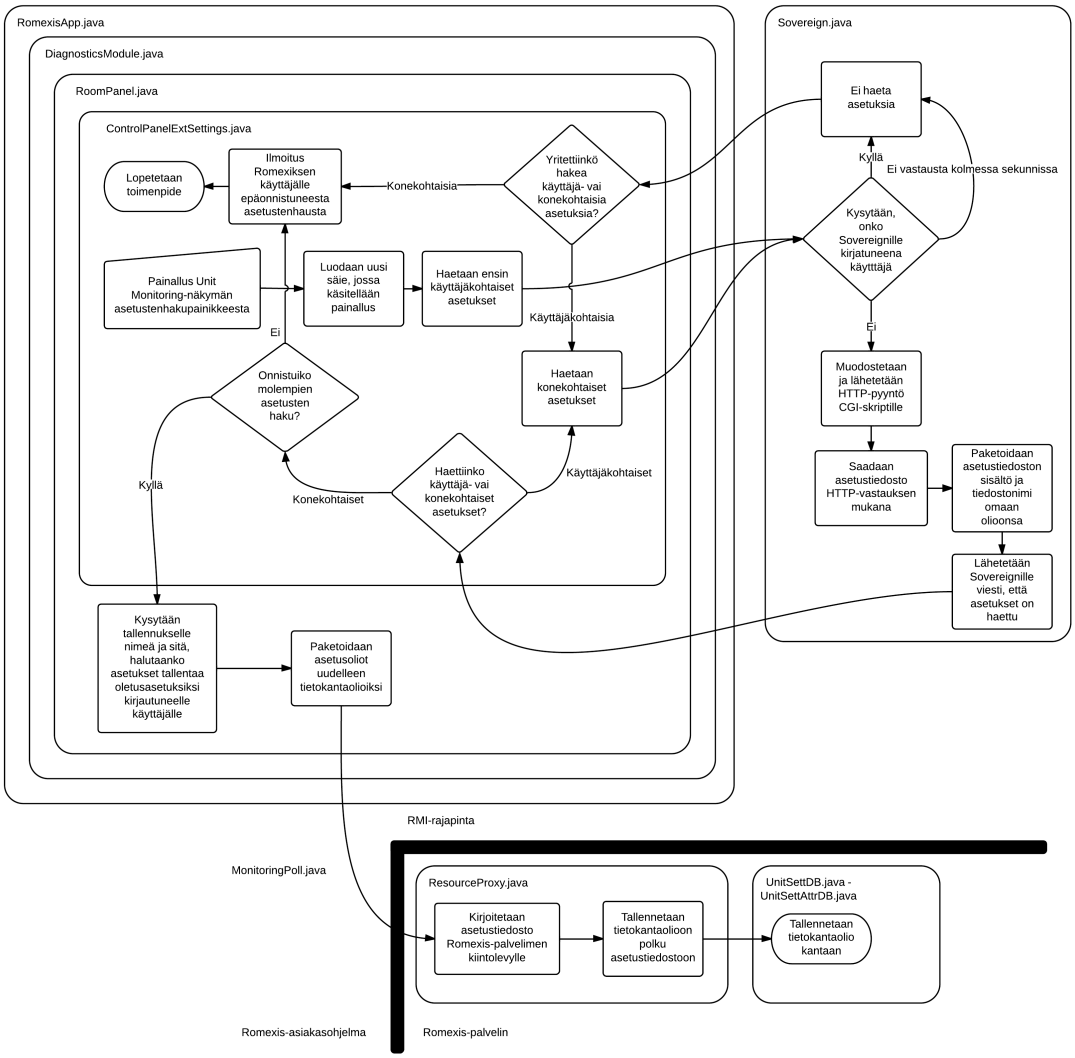
Asetusten hakemisen vaiheet Romexiksen ja sen käyttöliittymän kannalta on esitetty kuvassa 14. Kun asetustentallennusnappia painetaan, käyttöliittymä saa siitä tapahtuman. Tapahtuma käsitellään omassa säikeessään, joten Romexis-sovelluksen käyttöä voi jatkaa normaalisti. Tapahtumankäsittelyssä haetaan asetukset `Sovereign`-luokan kautta. Ensimmäiseksi haetaan käyttäjäasetukset. Aluksi hoitokoneelta varmistetaan, että asetukset voidaan hakea eli koneelle ei ole ketään kirjautuneena. Tämä tehdään lähettämällä viesti, jonka sisältö on `req.setting.transfer`. Romexis odottaa viestiin vastausta kolme sekuntia, jos vastausta ei tähän mennessä ole tullut, Romexis olettaa, että kone on käytössä, eikä jatka asetustenhakua. Sovereignin ACCU-kortti käsittelee Romexikselta tulleet viestit Planlink-prosessissaan. Sovereignin toiminta asetuksia haettaessa on esitetty yleisesti kuvassa 15. Romexiksen viestin saatuaan Planlink tarkistaa löytyykö ACCU:n tiedostojärjestelmästä tietty tiedosto, joka sinne luodaan, kun käyttäjä kirjautuu koneelle. Jos löytyy, tarkoittaa se, että hoitokoneelle on siis kirjautuneena käyttäjä, ja Planlinkin vastaus hoitokoneel-



Kuva 13: Asetustiedoston lähettämiseen käytetyn CGI-skriptin toiminta. [59]

le on, ettei asetuksia voida siirtää. Asetusten haku keskeytyy ja mitään ei tallenneta. Jos taas tiedostoa ei löydy, Planlink luo ACCU:lle oman tiedostonsa. Tiedoston olemassaolo estää käyttäjää kirjautumasta hoitokoneelle kesken asetustensiirron. Planlink vastaa Romexikselle viestillä, joka kertoo, että asetukset voidaan nyt siirtää. Romexis jatkaa asetustenhakua luomalla HTTP-pyynnön ja lähettämällä sen Sovereignille. Vastauksena Romexis saa siis tar-tiedoston, joka sisältää käyttäjäkohtaiset asetukset. Sovereignin HTTP-palvelin vaatii käyttäjänimen ja salasanan. Normaalisti nämä lähetetään vasta, kun palvelin on niitä pyytänyt, mutta tässä työssä ne lähetetään suoraan, sillä muuten palvelimen ja Romexiksen välistä HTTP-liikennettä ei saatu toimimaan. Asetusten vastaanottamisen jälkeen Romexis lähettää hoitokoneelle viestin, että asetustiedostot haettiin onnistuneesti, minkä jälkeen hoitokone poistaa kirjautumisen estävän tiedoston. Romexis taas jatkaa saamansa tiedoston tallentamisella. Tiedosto paketoidaan ensin Sovereignin omaan asetusluokkaan, jonka jäseniin tallennetaan tiedostonimi ja varsinainen tiedoston sisältö. Sama menettely toistetaan konekohtaisille asetuksille.

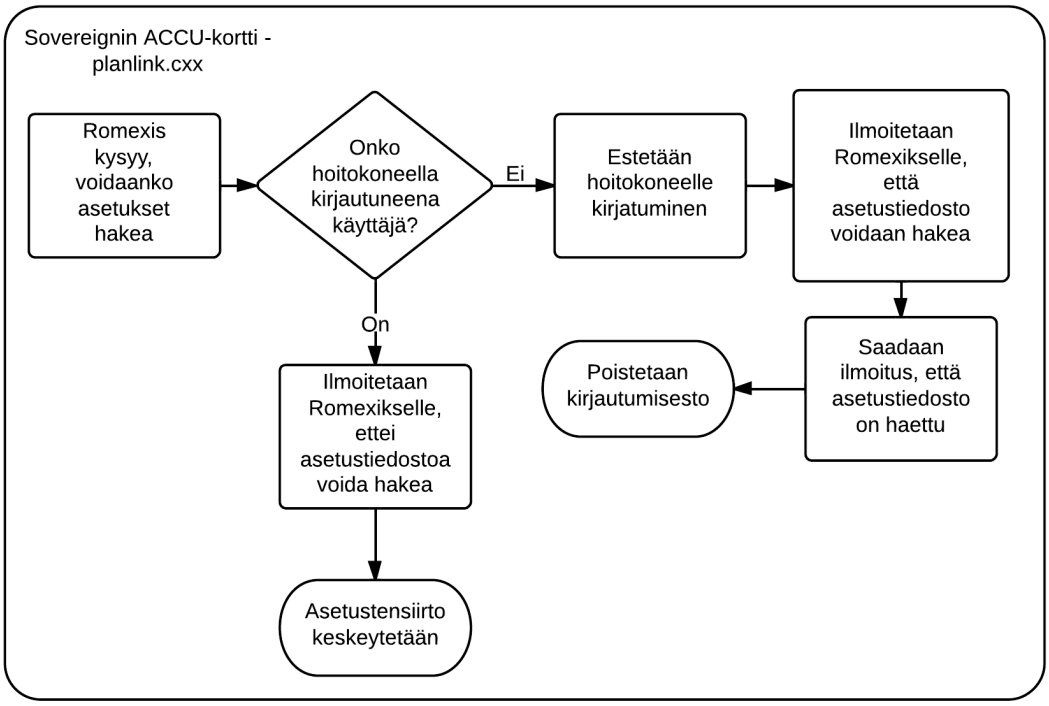
Käyttäjälle näytetään ponnahdusikkuna, jossa annetaan tallennukselle nimi ja valitaan, halutaanko asetukset tallentaa oletusasetuksiksi. Oletusasetukset tarkoittavat tässä tapauksessa sitä, että tallennetut käyttäjäasetukset lähetetään aina hoitokoneelle, kun kirjautuneena oleva käyttäjä kirjautuu Romexikseen, ja jos kyseiselle käyttäjälle on valittuna oletushoitokone, johon otetaan yhteys heti kirjautumisen yhteydessä. Hoitokoneen asetusolioiden sisältämä tieto paketoidaan tämän jälkeen uudelleen tietokantaolioihin. Romexis-asiakasohjelmalla on tiedossa muuttuja, joka viittaa RMI-rajapinnan kautta Romexis-palvelimella olevaan olioon. Tämän olion kautta kutsutaan metodia, jolla tietokantaolioiden sisältö saadaan tallennettua palvelimella pyörivään tietokantaan. RMI-rajapinta mahdollistaa siis asetustiedosto-



Kuva 14: Asetustiedostojen hakeminen `Unit Monitoring` -näkymästä. [59]

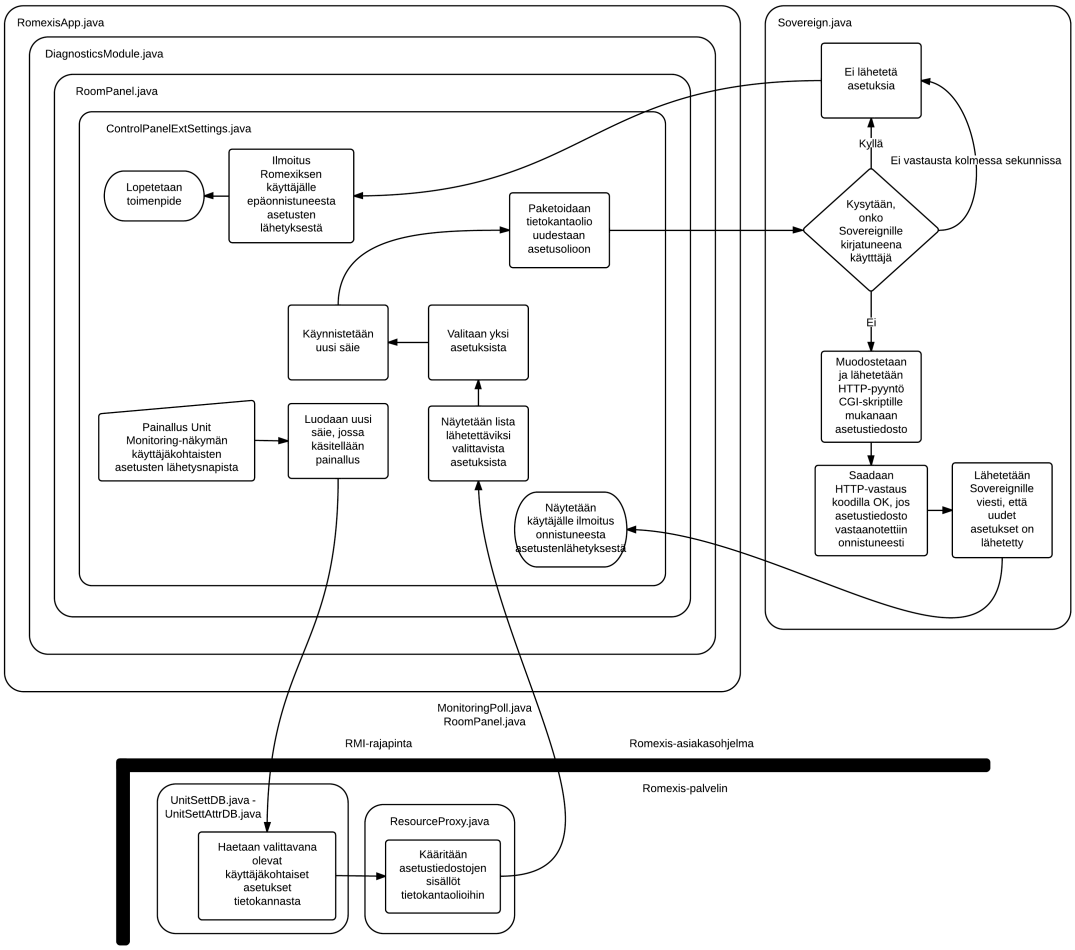
jen sisällön siirtymisen täysin automaattisesti asiakasohjelman ja palvelimen välillä. Palvelimella asetustiedoston sisältö kirjoitetaan koneen kiintolevylle käyttäjän oletuskotikansioon olion mukana tulleella tiedostonimellä. Absoluuttinen polku tiedostoon otetaan talteen tietokantaolioon bitteinä koodaten se `ISO-8859-1`-standardin mukaan, minkä jälkeen olion sisältö tallennetaan tietokantaan. Tiedostonimen bitit muutetaan tavuittain kymmenkantaisena tekstimuotoon, jokainen tavu erotetaan toisistaan puolipisteellä ja lopputulos tallennetaan tietokantaan tekstimuotoiseen kenttään.

Asetusten lähetys taas on siis mahdollista kahdesta eri paikasta. `Unit Monitoring`-näkymästä lähetetään pelkät käyttäjäasetukset. Tämä prosessi on esitetty kuvassa 16. Valittavana on pelkät kirjautuneena olevan Romexis-käyttäjän tallentamat asetukset. Tietokannasta haetuista tiedoista luodaan tietokantaoliot. Polku asetustiedostoon dekoodataan tietokannan taulun tekstikentästä ja tiedoston sisältö lue-



Kuva 15: Asetustiedostojen hakeminen Sovereignin näkökulmasta. [59]

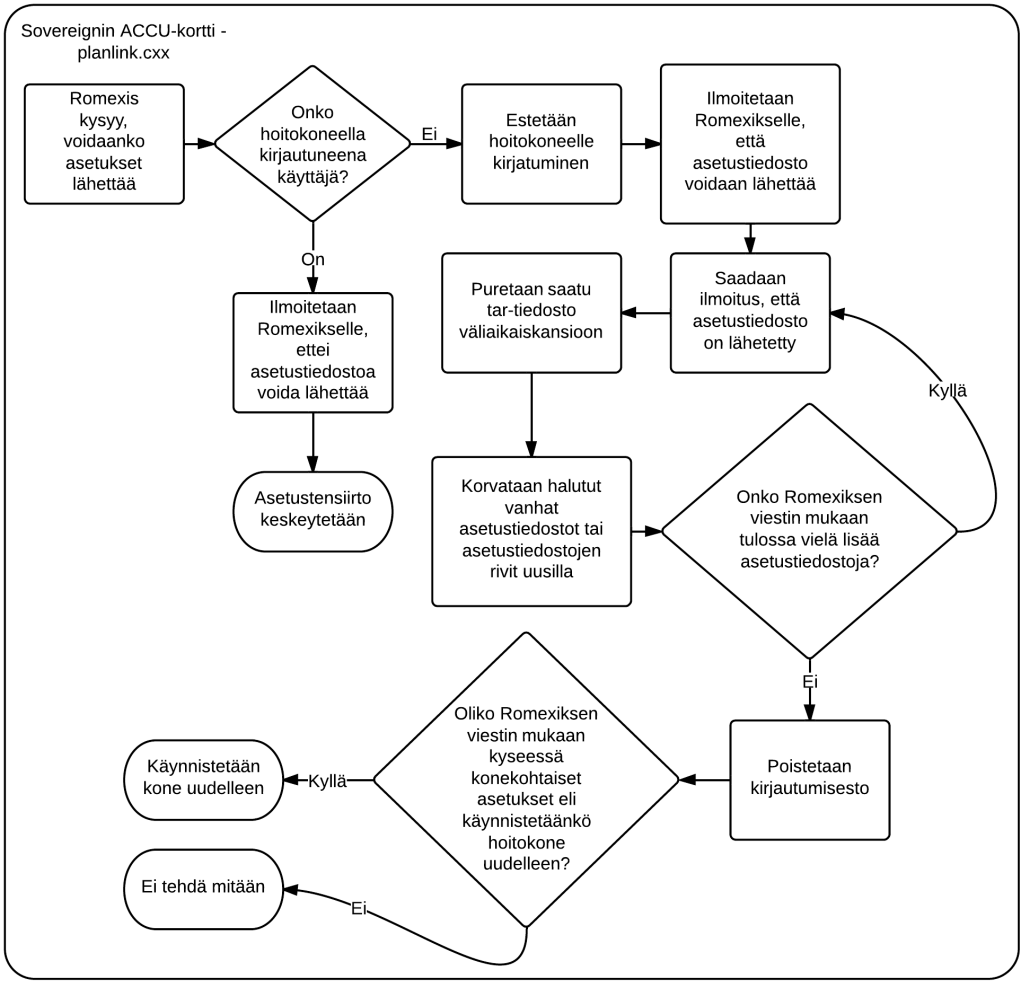
taan biteiksi tietokantaolion jäsenmuuttujaan. Tiedoston sisältö siirtyy näin taas automaattisesti RMI-rajapinnan yli palvelimelta asiakasohjelmalle. Kun lähetettävät asetukset on valittu, paketoidaan asetukset uudestaan Sovereignin oman asetusluokan olioon. Tämän jälkeen toimitaan kuten asetuksia haettaessa. Sovereignin näkökulmasta tämä asetusten vastaanottaminen on esitetty kuvassa 17. Hoitokoneelta varmistetaan, että kirjautuneena ei ole ketään ja lähetetään sitten HTTP POST -pyynnön sisällä asetustiedosto. Kun hoitokoneelta on saatu OK-vastaus, lähetetään hoitokoneelle viesti, jonka sisältö riippuu lähetetyn asetuksen tyypistä ja siitä, ollaanko vielä lähettämässä lisää asetuksia. Viesti vastaanotetaan jälleen ACCU:n Planlink prosessissa, ja se kertoo Sovereignille lähetettyjen asetusten tyypin, sen, tuleeko hoitokone käynnistää uudelleen viestin vastaanottamisen ja asetusten käsittelyn jälkeen, voiko kirjautumiseneston purkaa ja mikä on lähetetyn tiedoston nimi. Viestin vastaanottamisen jälkeen Planlink kutsuu skriptiä, joka purkaa ja kopioi tartiedoston sisällön oikeisiin paikkoihin. Jos ollaan siirtämässä konekohtaisia asetuksia, `unitconf.txt`-tiedoston sisältö käsitellään erikseen kutsumalla toista skriptiä, joka taas kutsuu vielä kolmatta skriptiä, joka hoitaa tärkeimmän asetustiedoston oikeiden rivien korvaamisen uuden tiedoston vastaavilla riveillä. Lisäksi vain osa konekohtaisten asetusten tiedostoista korvataan uusilla. Tämän toimenpiteen jälkeen hoitokone toimii siis Romexikselta saamansa viestin mukaisesti. Hoitokone käynnistetään uudelleen vain konekohtaisten asetusten lähettämisen jälkeen, sillä toisin kuin käyttäjäkohtaiset asetukset, uudet konekohtaiset asetukset tulevat käyttöön vasta



Kuva 16: Asetustiedostojen lähettäminen `Unit Monitoring` -näkymästä. [59]

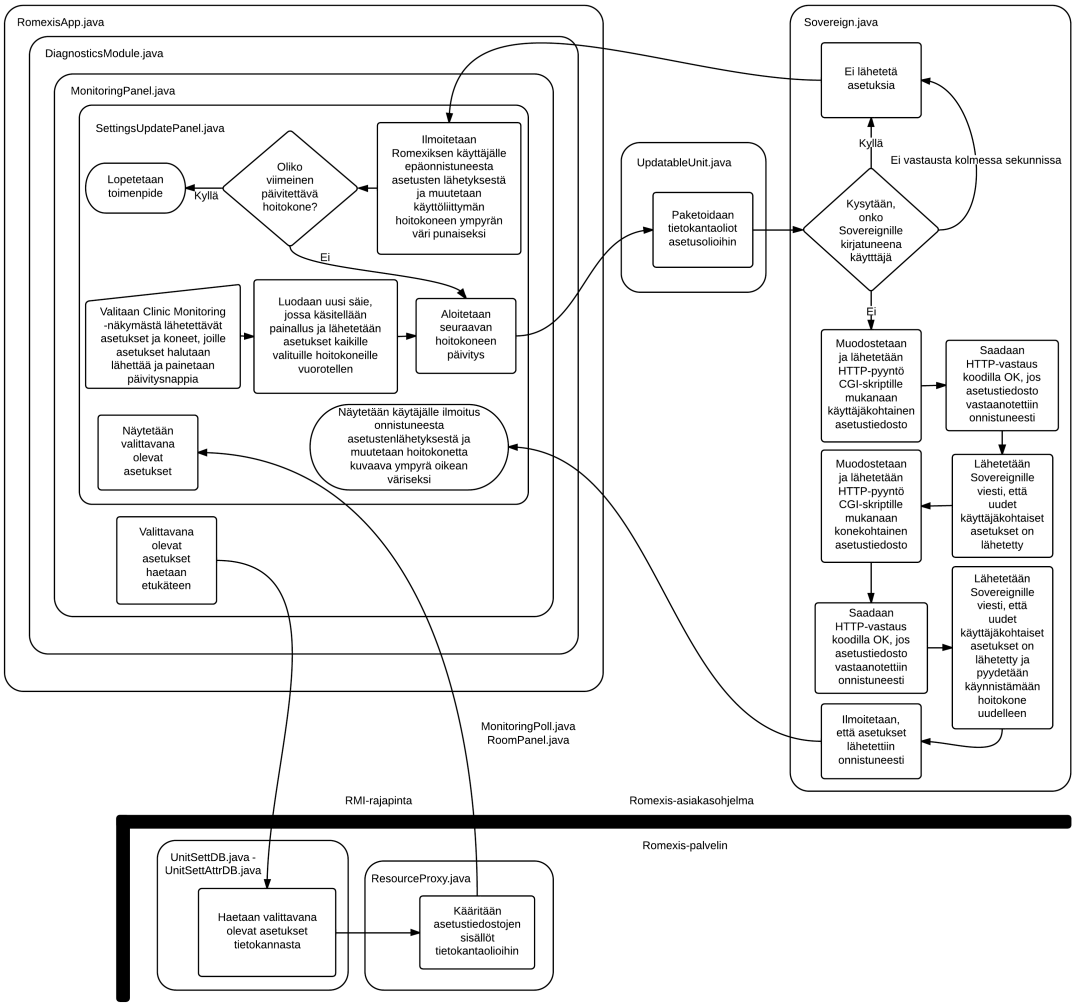
uudelleenkäynnistyksen jälkeen. Käyttäjäkohtaiset asetukset lähetetään siis ensin. Jos konekohtaisia ei olla lähettämässä, voidaan kirjautumisesto purkaa, muussa tapauksessa esto poistetaan vasta konekohtaisten asetusten lähettämisen jälkeen. Sovereign ei raportoi Romexikselle suorituksen etenemisestä, vaan Romexis ilmoittaa asetusten lähettämisen onnistuneen saatuaan vastauksen HTTP-pyyntöönsä. Vastaavasti ilmoitus annetaan tilanteessa, jossa asetuksia ei voida edes yrittää lähettää. On mahdollista että, jos jossain menee jokin pieleen kirjautumiseston ollessa päällä, jää esto purkamatta, ja hoitokone täytyy käynnistää uudelleen, jotta sitä voitaisiin käyttää. Ainakin virheen tapahtuessa Romexiksen päässä tulisi tästä lähettää viesti Sovereignille, joka purkaisi lukituksen. Toisaalta voi vika olla yhtä hyvin nimenomaan tämän viestin lähetyksessä tai sen käsittelyssä. Joka tapaukessa samanlaista asetustensiirron tilan palkkia kuin Compactilla ei toteutettu. Se ei kyllä voisikaan toimia samalla tavalla Sovereignin kohdalla, jossa tiedoston lähettäminen ei kestä sekuntiakaan, kun taas Compactilla asetusbittien siirtäminen ja kirjoittaminen hoitokoneen muistiin vastaa isoa osaa asetustensiirtoon kuluvasta ajasta.

`Clinic Monitoring`-näkymästä asetustenlähetys tapahtuu hyvin pitkälle samal-



Kuva 17: Asetustiedostojen vastaanottaminen Sovereignin näkökulmasta. [59]

la tavalla. Tästä löytyy havainnollistava kaavio kuvasta 18. Pelkkien käyttäjäasetusten sijaan, lähetetään myös konekohtaiset asetukset, jolloin hoitokone uudelleenkäynnistetään toimituksen jälkeen. Näkymästä voidaan valita useita hoitokoneita, joille samat asetukset lähetetään. Päivitysnappia painettaessa aloitetaan uusi säie, jossa asetukset lähetetään vuorotellen kullekin hoitokoneelle. Lähettämisen onnistuessa tai epäonnistuessa lähetetään tapahtuma, joka muuttaa hoitokonetta edustavan ympyrän vihreäksi tai punaiseksi. Kunkin hoitokoneen lopputuloksesta näytetään myös ponnahdusikkuna. Samasta näkymästä voidaan vain kyseiselle Romexis-käyttäjälle näkyvät asetukset muuttaa globaaleiksi kaikille käyttäjille näkyviksi asetuksiksi ja toisinpäin. Lisäksi on mahdollista `Unit Monitoring`-näkymästä poistaa yksittäisiä asetuksia tallennusnimen perusteella. Käyttäjä- ja konekohtaiset poistetaan aina samalla kertaa. Tietokantamerkinnän lisäksi poistetaan myös varsinaiset asetustiedostot. [59]



Kuva 18: Asetustiedostojen lähettäminen `Clinic Monitoring` -näkymästä. [59]

4.3 Ominaisuuden testaus

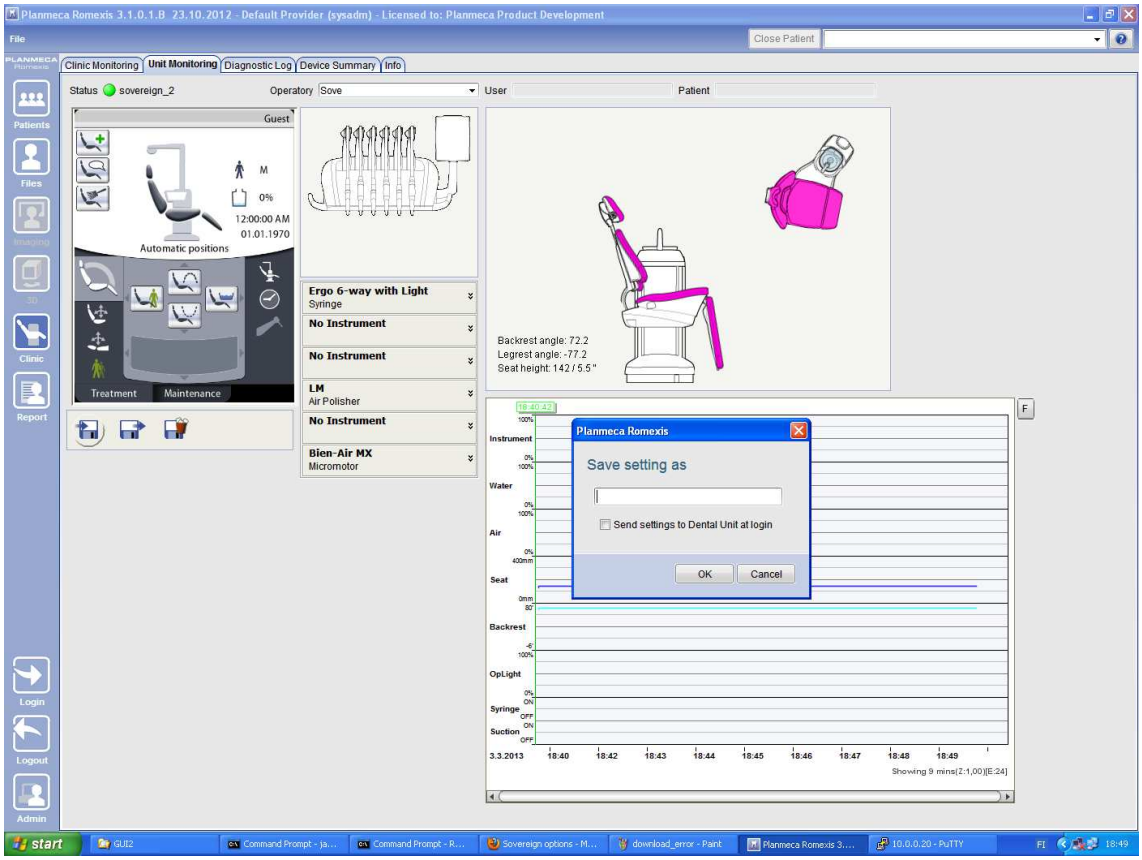
Työssä toteutetulle ominaisuudelle ei ole vielä määritetty omia testikäytäntöjään vaan käytetään samoja testejä kuin Compactin vastaavalle ominaisuudelle. Sovereignille ominaisuudessa olisi enemmän testattavaa, joten testejä olisi syytä kehittää. Testattavat asiat on listattu taulukoissa 4 ja 5. Testausta tehdään erikseen Romexiksen ja hoitokoneen testauksen yhteydessä.

Taulukko 4: Asetustensiirto-ominaisuuden testaus Sovereignin testauksen yhteydessä.

Vaihe	**Toiminta**	**Odotettu lopputulos / Hyväksytty kriteeri**
1.	Avaa yhteys	
2.	Mene hoitokoneen asetuksiin	
3.	Valitse hoitokoneet uusien asetusten kopioimiseksi	Hoitokoneet valittu
4.	Valitse asetustiedosto	Asetustiedosto valittu
5.	Paina asetusten päivitysnappia	Asetukset päivitettiin onnistuneesti

Taulukko 5: Asetustensiirto-ominaisuuden testaus Romexiksen testauksen yhteydessä.

Vaihe	**Toiminta**	**Odotettu lopputulos / Hyväksytty kriteeri**
1.	Valitse hoitokone päivitettäväksi. Yritä valita hoitokone sekä pohjapiirroksesta että hoitokonelistasta	Hoitokone valikoituu oikein molemmista paikoista
2.	Valitse lähetettävät asetukset	Valinta onnistui
3.	Päivitä valitun hoitokoneen asetukset	Päivitys onnistui
4.	Toista testi valiten useita hoitokoneita samalla kertaa (esimerkiksi 2, 10, 100)	Kaikkien hoitokoneiden päivitys onnistui
5.	Yritä lähettää Compactin asetukset Sovereignille	Asetusten lähetys ei onnistunut
6.	Yritä päivittää samalla kertaa Compact ja Sovereign -hoitokoneita	Päivitys ei onnistu. Käyttöliittymä ilmoittaa virheestä informatiivisesti.
7.	Yritä päivittää hoitokonetta, jolle on kirjautuneena käyttäjä	Päivittäminen ei onnistu
8.	Tallenna asetuksia vähintään kahdelle eri Romexis-käyttäjälle	Tallennus onnistui
9.	Vahvista, että kukin käyttäjä näkee tallentamansa ja vain itse tallentamansa asetukset valittavien listassa	Asetukset näkyvät
10.	Muuta yksi ensimmäisen käyttäjän asetuksista globaaliksi	Muutos onnistuu
11.	Vahvista, että muutettu asetus näkyy myös toisella käyttäjällä	Asetus näkyy listassa
12.	Yritä muuttaa toisella käyttäjällä globaalia asetusta takaisin henkilökohtaiseksi	Muutos ei onnistu
13.	Toista kohta 12. ensimmäiselle käyttäjälle	Muutos onnistui
14.	Toista kohta 9. molemmille käyttäjille	Kumpikin käyttäjä näkee vain omat asetuksensa

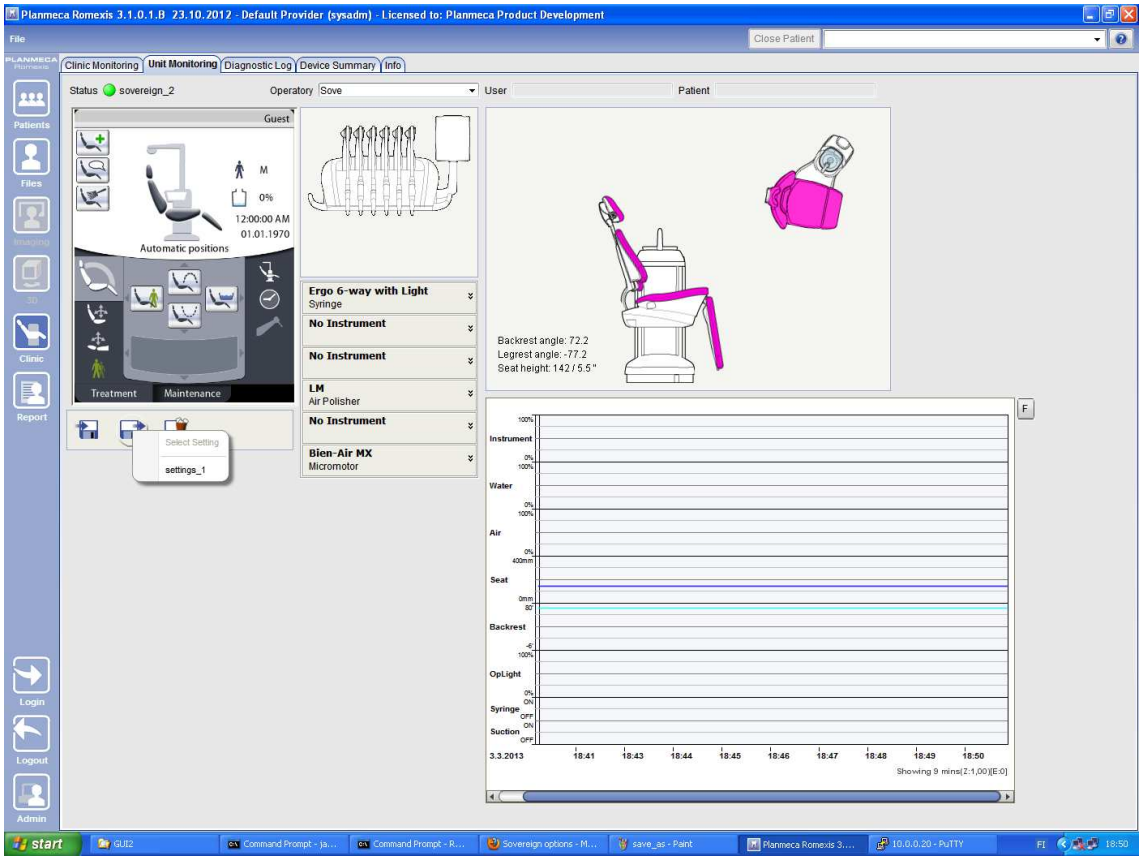


Kuva 19: Asetustentallennusikkuna `Unit Monitoring` -näkymässä.

5 Tulosten esittely

Kuvassa 19 näkyy Romexiksen `Unit Monitoring` -näkymä, josta voidaan ohjata ja tarkkailla yhtä hoitokonetta. Hoitokoneen käyttöliittymän näyttävän vasemmalla olevan ikkunan alta löytyy kolme painiketta, joilla asetustensiirto-ominaisuutta käytetään. Ensimmäisestä painikkeesta haetaan kerralla yhden hoitokoneen kone- ja käyttäjäkohtaiset asetukset. Kun asetustiedostot on saatu käyttäjälle näytetään kuvassa näkyvä ponnahdusikkuna, joka kysyy nimeä, jolla asetukset halutaan tallentaa. Lisäksi ikkunasta valitaan, halutaanko tallennattavana olevat käyttäjäasetukset tallentaa valittuna olevan hoitokoneen oletusasetuksiksi, jotka lähetetään hoitokoneelle aina, kun Romexiksen nykyinen käyttäjä kirjautuu. Asetukset tallennetaan automaattisesti vain kirjautuneena olevalle käyttäjälle käytössäoleviksi.

Toisesta painikkeesta voidaan lähettää hoitokoneelle halutut käyttäjäasetukset. Kun napista painetaan, avautuu kuvassa 20 näkyvä lista valittavana olevista asetuksista. Listassa on vain kirjautuneena olevan käyttäjän tallentamat asetukset. Globaaleiksi muutettuja asetuksia ei tällä hetkellä näytetä. Jos lähettäminen ei onnistu, koska hoitokoneelle on kirjautuneena käyttäjä tai jostain muusta syystä, näytetään käyttäjälle kuvan 21 ikkunaa vastaava viesti. Samanlainen viesti näytetään myös asetustenhaun epäonnistuessa. Vastaavasti näytetään kuvan 22 mukainen viesti, kun asetusten lähetys onnistui. Kolmannesta painikkeesta voidaan poistaa tallennettuja

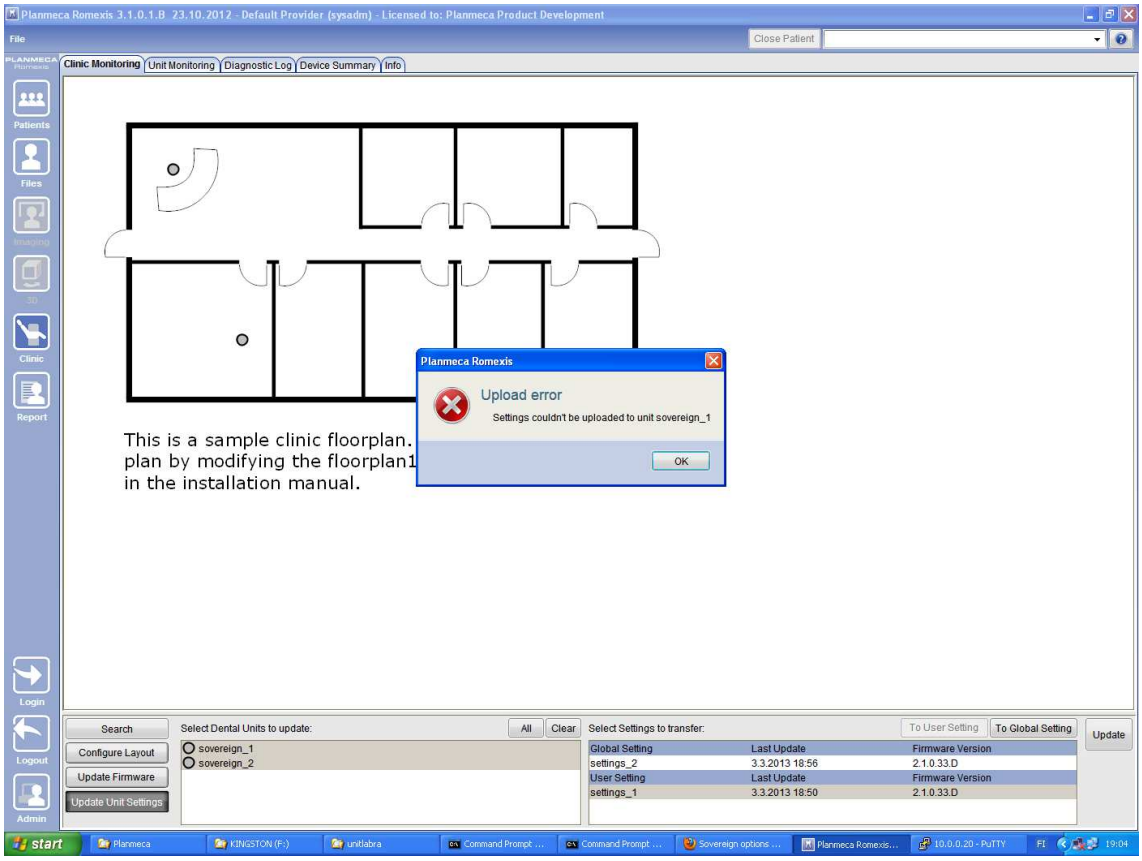


Kuva 20: Lähetettävien asetusten valinta `Unit Monitoring` -näkymässä.

asetuksia. Napin painalluksesta avautuu samanlainen lista kuin tallennuksia lähetettäessä. Poistettava asetus valitaan ja poiston onnistuessa se katoaa listasta.

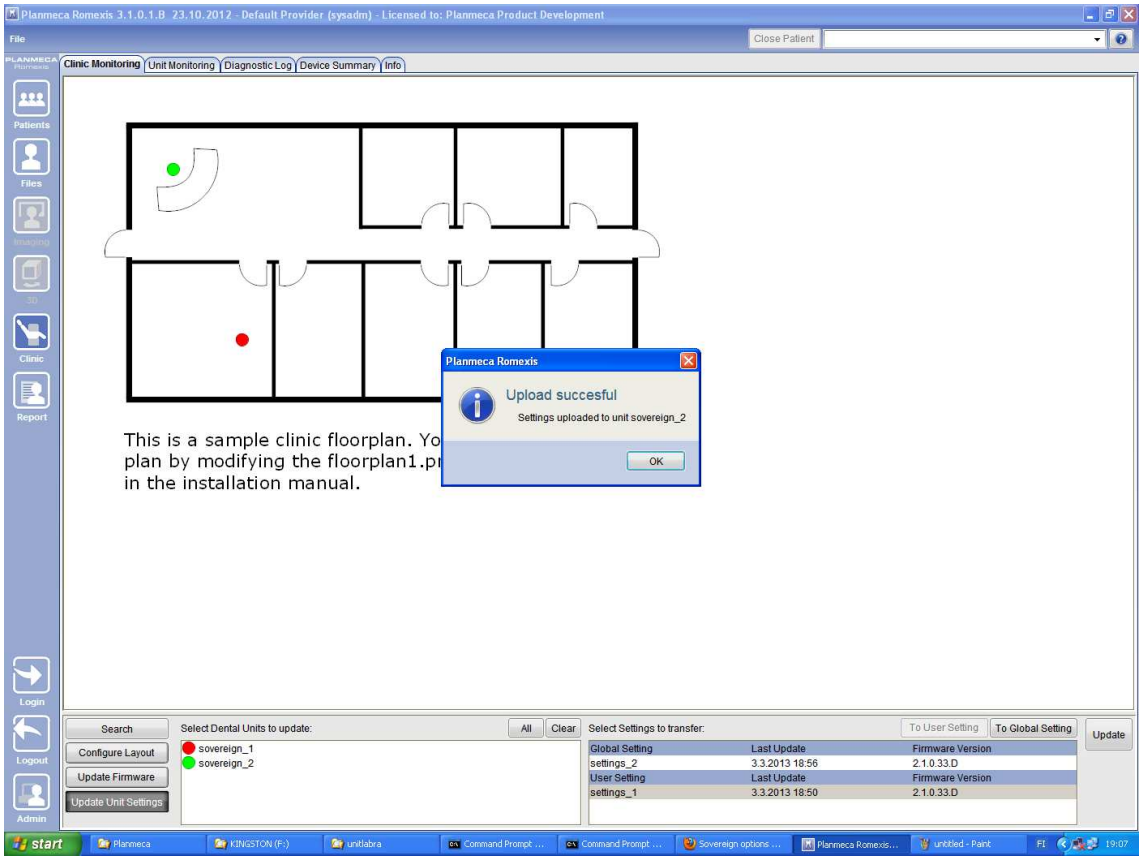
Kuvassa 21 on Romexiksen `Clinic Monitoring` -näkymä. Siinä näkyy esimerkki klinikan pohjapiirroksesta, johon hoitokoneet on sijoitettu. Alhaalta on valittu `Update Unit Settings`-välilehti, josta voidaan valita useita hoitokoneita kerralla ja lähettää niille kaikille samat asetukset. Oikealla näkyy lista valittavista asetuksista, jotka on ryhmitelty sen mukaan ovatko ne globaaleja vai vain senhetkiselle käyttäjälle näkyviä. Käyttäjä voi muuttaa omat asetuksensa globaaleiksi ja takaisin henkilökohtaisiksi. Globaalien muuttaminen henkilökohtaisiksi on mahdollista siis vain sellaisille asetuksille, jotka ovat alunperin kyseisen käyttäjän tallentamia. Romexis kertoo käyttäjälle epäonnistuneista ja onnistuneista asetustensiirroista kuvissa 21 ja 22 näkyvin ilmoituksin. Listauksessa ja pohjapiirroksessa hoitokoneiden vierellä olevien ympyröiden värit muuttuvat myös tuloksen mukaan vihreiksi tai punaisiksi kuvassa 22 näkyvällä tavalla.

Kuvassa 23 on sivu, joka syntyy HTML-koodista, jonka työssä toteutettu konekohtaisten asetusten lähettämiseen tarkoitettu CGI-skripti palauttaa. Lomakkeella voidaan lähettää hoitokoneelle mikä tahansa tiedosto. Se on toki tarkoitettu konekohtaisen asetustiedoston lähettämiseksi. Lomaketta ja CGI-skriptin toimintalogiikkaa on toki tarpeen kehittää eteenpäin, mutta sivu, yhdesssä käyttäjäkohtaisten ase-

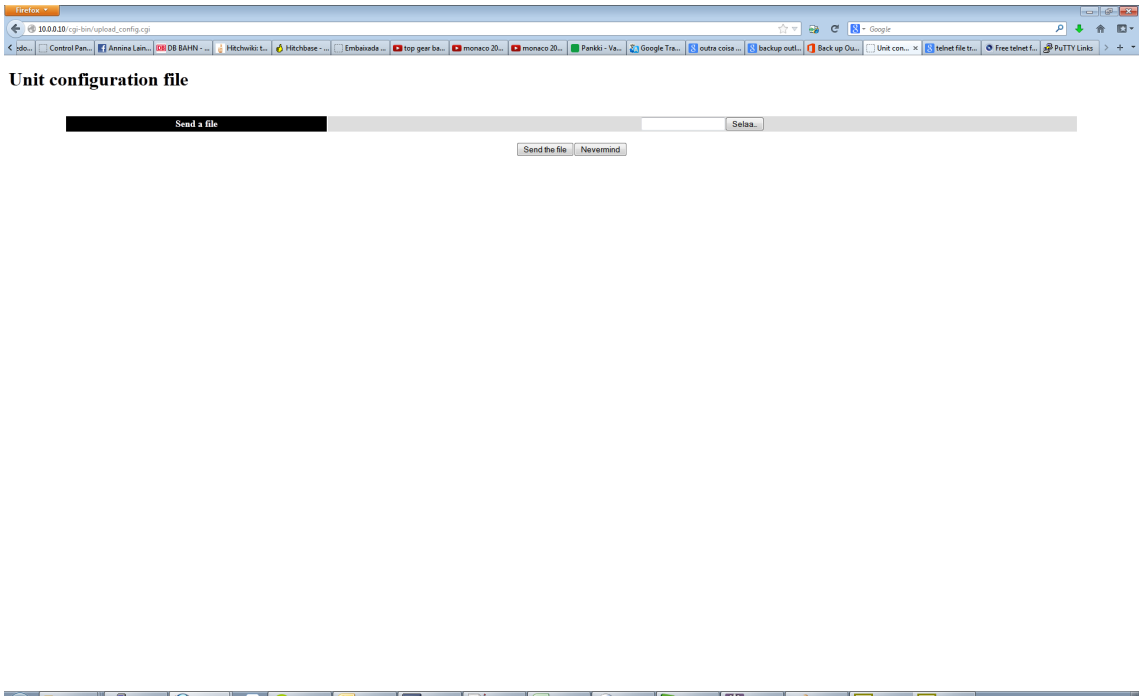


Kuva 21: Ilmoitus virheestä asetustenlähetyksessä `Clinic Monitoring` -näkymässä.

tusten lomakkeen kanssa, olisi tarkoitus lisätä osaksi Sovereignin CGI-teknologialla toteutettua web-käyttöliittymää, jolloin Sovereignin uudet asetukset voitaisiin ainakin lähettää, jos ei hakea, selaimen kautta. [5]



Kuva 22: Ilmoitus onnistuneesta asetustenlähetyksestä `Clinic Monitoring` -näkymässä.



Kuva 23: Lomake konekohtaisten asetusten lähettämiseksi Sovereignille.

6 Tarkastelu

6.1 Arviointi

Kokemuksia työssä tehdyn ominaisuuden toimimisesta ei ehditty saada kentältä, sillä se ei ehtinyt käyttöön asti. Eikä palautetta myöskään Compactin vastaavasta ominaisuudesta ollut saatavilla. On kuitenkin täysin selvää, että ominaisuus on erittäin hyödyllinen. Sen toiminnan todettiin myös olevan hyvä ja luotettava kehityksen aikana ja sen jälkeisissä testeissä. Siinä mielessä voidaan todeta työn onnistuneen ja tavoitteiden tultua saavutetuiksi. Ominaisuus täyttää sille asetetut vaatimukset toteuttaen saman toiminnallisuuden kuin vastaava Compact-hoitokoneen ominaisuus. Tähän työhön sisällytetyllä kehityksellä päätettiin lopulta pyrkiä juuri siihen. Ominaisuus toimii hyvin toteuttaen tarkoituksensa. Kehityksen ja sen suunnittelu aikana saatiin kuitenkin runsaasti ideoita siitä, miten ominaisuutta voidaan jatkokehittää. Kokemuksia ja palautetta kentältä tulisi myös jatkossa yrittää saada esimerkiksi haastattelujen ja kyselyjen kautta lisäkehitysideoiden synnyttämiseksi ja ominaisuuden onnistumisen lopulliseksi toteamiseksi.

6.2 Ominaisuuden jatkokehitys

Asetusten siirto-ominaisuuden parantamiseksi on useita jatkokehitysideoita. Asetustensiirron voisi tehdä mahdolliseksi myös hoitokoneen päästä. Tällöin lääkäri voisi aina hoitokoneelle tullessaan hakea siihen Romexis-palvelimelle tallentamansa asetukset. Käytön jälkeen ja tehtyään asetuksiin muutoksia lääkäri voisi taas tallentaa muutokset palvelimelle, jolloin ne olisivat käytössä kaikilla muillakin klinikan koneilla, eikä olisi väliä, millä klinikan koneella lääkäri seuraavaksi toimisi. Tämä olisi siis erityisen hyödyllistä klinikoilla, joilla lääkärit käyttävät päivän aikana useita eri hoitokoneita ja toisaalta joilla samaa konetta käyttävät useammat eri lääkärit. Enää ei lääkärin tarvitsisi välttämättä kantaa mukanaan henkilökohtaisia asetuksiaan USB-tikulla. Hoitokone voisi myös automaattisesti lääkärin kirjautuessa sisään hakea oletusasetuksiksi merkityt asetukset, jolloin suurimmassa osassa käyttötapauksista käyttäjältä ei vaadita mitään toimenpiteitä. Vastaavasti oletusasetukset tallentuisivat automaattisesti uloskirjautumisen yhteydessä. Asetustenhallinnan käyttäminen hoitokoneesta käsin vähentäisi myös Romexis-asiakasohjelmien ja sitä kautta hoitokoneiden vieressä olevien chair side -PC:den käyttötarvetta. Kaikki tarpeellinen kommunikaatio tapahtuisi hoitokoneen ja palvelimen välillä.

On harkittu myös, että toteutettaisiin asetusten muokkausominaisuus asiakasohjelmaan. Tämän toteuttamiseen löytyy useampia vaihtoehtoja. Jo tällä hetkellä Romexiksessa pyörii Sovereignin graafista käyttöliittymää vastaava sovellus. Sovellus ei vielä ole täysin replikoitu vastaamaan Sovereignin ja sen käyttöliittymän tilaa. Jos PC:llä pyörivän käyttöliittymän ja Sovereignin välinen kommunikaatio ja replikointi täydennettäisiin, voitaisiin Sovereignin asetuksia muokata tämän käyttöliittymän kautta aivan kuten itse hoitokoneen käyttöliittymälläkin. Toisaalta Romexikseen voitaisiin toteuttaa asetuksia varten oma käyttöliittymä, joka lähettäisi muutokset Sovereignille. Tämä toteutus voisi viestinnältään joko emuloida oikeaa käyttöliitty-

mää tai käyttää tekstipohjaisia viestejä kuten muussa Sovereignin ja Romexiksen välisessä viestinnässä. Käytännössä neljäs vaihtoehto olisi, että Romexiksen käyttöliittymän kautta muokattaisiin suoraan asetustiedostoja, jotka sitten lähetettäisiin hoitokoneelle. Samaan yhteyteen voisi toteuttaa ominaisuuden, jolla kaikki asetukset asetetaan takaisin oletus- tai tehdasasetuksiin. Oletusasetukset olisivat tallessa joko hoitokoneella tai sitten Romexis-PC:llä joko konekohtaisina tai ei.

Toteutetun ominaisuuden testaajien palautteen perusteella, olisi hyödyllistä, jos lähetettäessä samoja asetuksia useammalle koneelle samalla kertaa, Romexis tarkastaisi ennen lähetystä kaikkien valittujen koneiden kirjautumistilanteen ennen yksienkään asetusten lähettämistä. Nykyisellä toteutuksella kaikki asetuksien lähetykset tapahtuvat samassa säikeessä. Tällöin, jos hoitokoneita on valittuna vaikkapa sata, joista viiteenkymmeneen lähetys ei onnistu, on hankalaa ensin odottaa pitkään, että lähetykset ensin joko onnistuvat tai epäonnistuvat, merkitä ylös, mihin koneisiin lähetys ei onnistunut, käydä kirjautumassa käyttäjä ulos näiltä koneilta ja tehdä lähetys uudelleen näille koneille. Jos kirjatumistarkistus tehtäisiin heti kaikille koneille, vältyttäisiin turhalta odottamiselta ja mahdolliselta turhalta asetusten uudelleenlähettämiseltä. Nykyisellä toteutuksella tällaiseen pääseminen on hieman hankalaa, kun kaikki lähetykset tehdään peräkkäin samassa säikeessä. Lähetys pitäisi muuttaa kaksiportaiseksi, jossa ensin yhdessä säikeessä tehtäisiin tarkistus ja itse lähetys vasta sitten omassa säikeessään.

Romexiksen käyttöliittymässä on hoitokoneiden asetuksiin littyvä terminologia hieman sekavaa ja harhaanjohtavaa. Hoitokoneen asetukset erotellaan käyttäjäasetuksiin ja konekohtaisiin asetuksiin. Toisaalta asetukset erotellaan käyttäjäasetuksiin ja globaaleihin asetuksiin sen mukaan, ovatko asetukset käytössä vain kyseessä olevalla Romexis-käyttäjällä vai kaikilla käyttäjillä. Tässä menee siis hoitokoneiden käyttäjät ja Romexis-käyttäjät sekaisin. Terminologiaa voisi selkeyttää ja muuttaa tekstejä Romexiksen käyttöliittymästä. Lisäksi sekavuuttaa aiheuttaa se, että asetustensiirtotoiminnallisuutta löytyy kahdesta eri paikasta. Hoitokonenäkymästä voidaan hakea koneen kaikki asetukset, mutta sieltä voidaan lähettää vain käyttäjäkohtaiset asetukset. Tämä ei ilmene käyttöliittymästä selkeästi. Vain klinikkanäkymästä voidaan lähettää sekä käyttäjä- että koneasetukset yhdellä kertaa. Toisaalta sieltä ei voi hakea tai poistaa asetuksia laisinkaan. Myös käyttöohjeen päivitys auttaisi näihin ongelmiin. Jostain syystä globaaleiksi muutetut asetukset ovat käytössä kaikilla käyttäjillä vain klinikkanäkymässä. Hoitokonenäkymässä niitä ei näy. Käyttöliittymää voisi myös yleisesti selkeyttää ja sen ulkonäköä nykyaikaistaa, mutta tämä olisikin sitten osa isompaa koko Romexiksen käsittävää remonttia. Romexiksen käytettävyyttä voisi tutkia ja miettiä ihan perusteellisesti.

Ominaisuuden väärinkäytön vaikeuttamiseksi ja virheensiedon lisäämiseksi tulisi prosessiin lisätä oikeellisuustarkistus lähetetyille tiedostoille. Tämä olisi parempi tehdä hoitokoneeseen, joka tietäisi aina itse, millaisia tiedostoja sen ohjelmistoversio ja kokoonpano vaatii ja mitä tiedostojen pitäisi sisältää. Romexiksessa säästyttäisiin kaikkien eri versioiden tietojen ylläpitämiseltä. Yksinkertaisimmillaan tämä voisi toki tarkoittaa vain sitä, että Romexis vertaisi Romexiksen ja asetustietojen versionumeroita ja sillä olisi tieto siitä, mitkä versiot ovat keskenään yhteensopivia. Nyt on tosin mahdollista lähettää käytännössä mitä tahansa sisältäviä tiedostoja, kunhan

tiedostonimet ovat oikeat. Sovereign ylikirjoittaa tällöin suoraan vanhat tiedostot ja kaikki asetukset menetetään. Versionumerojen ja löytyvien tiedostojen tarkistuksen lisäksi pitäisi siis tutkia tiedostojen sisältöjä. Virheellisiä rivejä voisi korjata ja puuttuvia rivejä korvata jollain oletusarvoilla, mitä tehdäänkin jo joidenkin tiedostojen kohdalla. Jos lähetetyissä asetuksissa on jotain vikaa, esimerkiksi jokin tiedosto puuttuu, tai muuten jokin epäonnistuu asetusten lähettämisen jälkeen, siitä ei mene mitään tietoa Romexikselle. Olisi hyvä lisätä keskusteluun jokin viesti, joka kertoisi siirron ja uusien asetusten käyttöönoton onnistumisesta.

Sovereignin omaa sisäistä asetustenhallintaa voisi toisaalta myös kehittää ja yksinkertaistaa. Voisi miettiä, onko tarpeen, että asetukset on jaettu niin moneen eri tiedostoon. Toisaalta on hölmöä, että samassa tiedostossa on siirrettäviä ja eisiirrettäviä asetuksia, kun hoitokoneen kalibrointiasetuksia ja muita tietyn koneen asetuksia on samassa tiedostossa esimerkiksi käyttöliittymään liittyvien valintojen kanssa. Voisi esimerkiksi yrittää mahduttaa yhteen tiedostoon kaikki siirrettävät konekohtaiset asetukset ja toiseen kaikki käyttäjäkohtaiset asetukset. Tämä vaatisi ehkä myös sitä, että yksinkertaistettaisiin tapaa, jolla käytössä olevien käyttäjäkohtaisten asetusten valinta on toteutettu. Tähän liittyy myös tulevaisuudessa toteutettavien hoitotilojen huomioonottaminen. Miettiä voisi myös sitä, mitkä asetukset lasketaan konekohtaisiin asetuksiin. Hyvinkin esimerkiksi käyttöliittymään tehdyt muutokset voisivat siirtyä käyttäjäasetusten mukaan. Ehkä kaikkia asetuksia voisi kohdella tällä tavalla.

Ominaisuuden käytettävyyttä ja hyödyllisyyttä voisi parantaa huomattavasti yhdistämällä tiiviimmin toisiinsa hoitokoneen käyttäjän, Romexiksen käyttäjän ja kunkin käyttäjäasetustiedoston omistajan. Tätä voisi jopa laajentaa koskemaan erikseen hoitokonetta käyttävää lääkäriä ja hoitajaa. Myös USB:lla siirrettävät asetukset pitäisi yhdistää käyttäjään. Vaihtoehtona olisi myös yhdistäminen tiettyyn hoitokoneeseen. Jo nyt toki tallennetaan hoitokoneen nimi kantaan, mutta tämä ei varsinaisesti yksilöi hoitokonetta. Tämä kaikki ainakin helpottaisi eri tallennettujen asetusten hallintaa. Lisäksi tämä mahdollistaisi aiemmin mainitun asetustenhallinnan käytön hoitokoneesta käsin. Käyttäjäasetuksia voisi palauttaa yksittäin eikä vain kaikkia kerralla. Tämä toki vaatisi, että asetukset erotettaisiin haettaessa eri tiedostoihin. Tällä tavoin päästäisiin tavallaan myös eroon hoitokoneen viiden käyttäjän rajoituksesta. Konekohtaiset asetukset voisi samalla tehdä lähetettäviksi ilman käyttäjäkohtaisia asetuksia.

On myös joitakin hieman enemmän yksityiskohtiin menevämpiä kehitysideoita. Kun asetuksia lähdetään siirtämään, hoitokoneelle kirjautuminen estetään. Jos jokin asetustensiirrossa menee pieleen tai jos joku viesteistä ei mene perille, saattaa käydä niin, että hoitokoneen lukitus jää päälle, ja kone on käynnistettävä uudelleen sen käyttämiseksi. Lukitukseen voisi laittaa ajastimen, jonka jälkeen lukitus viimeistään avattaisiin. Toisaalta viestintää hoitokoneen ja Romexiksen välillä pitäisi lisätä, jotta mahdolliset epäonnistumiset olisivat molempien ohjelmien tiedossa ja niistä osattaisiin palautua. Tähän liittyen myös Sovereignin ilmoitusta, kun lukitulle koneelle yrittää kirjautua, voisi selkeyttää.

Asetustensiirto ei siis onnistu, jos koneelle on kirjautuneena käyttäjä. Tällaisessa tilanteessa saattaa tulla työlääksi käydä läpi kaikki koneet uloskirjaten käyttäjät.

Uloskirjauksen pakotuksen voisi siis mahdollistaa tehtäväksi etänä. Pitäisi vain varmistua, ettei konetta ole kukaan todella käyttämässä. Jos koneelle lähetetään konekohtaisia asetuksia, ne lähetetään viimeisenä, ja sen jälkeen hoitokone käynnistetään uudelleen automaattisesti. Tässä kohtaa voisi harkita, että käynnistys pitäisi hyväksyä hoitokoneelta. Toisaalta tämä jälleen tekisi pahimmillaan päivityksestä erittäin työlästä, eikä käynnistyksen pitäisi haitata, kun koneella ei ole käyttäjää. Toinen mahdollisuus olisi vain lisätä varoitus, että kone käynnistetään uudelleen, ja antaa käyttäjälle vaikka 30 sekuntia peruuttaa tämä.

Ominaisuuden käyttämistä Romexiksesta yksinkertaistaisi, jos sen etenemistä voisi seurata paremmin. Tämä koskee lähinnä sitä, kun lähetetään kerralla asetukset suurelle määrälle hoitokoneita. Jälleen viestintää hoitokoneen ja Romexiksen välillä tulisi lisätä. Tällä hetkellä onnistumiseksihan katsotaan, jos tiedosto saadaan siirtymään. Tällä hetkellä asetukset siirretään yhdessä säikeessä peräkkäin. Kunkin hoitokoneen siirtäminen omaan säikeeseensä auttaisi tähänkin ongelmaan. Epäonnistuneisiin asetustenlähetyksiin voisi puuttua viipymättä, eikä tarvitsisi odottaa, että kaikki hoitokoneet käytäisiin loppuun ensin.

Lisää kehitystyötä vaatii ominaisuuden suunniteltu käyttö selaimesta osana CGI:llä toteutettua web-käyttöliittymää. Asetustensiirtoon tehdyt CGI-skriptit vaativat muutoksia tämän mahdollistamiseksi. Skripteillä voi kyllä hakea asetukset hoitokoneelta ja lähettää hoitokoneelle tiedoston, mutta mitään viestejä ei tällöin kulje, joten esimerkiksi hoitokoneen kirjautumistilannetta ei tarkasteta ja lähetetystä asetustiedostosta ei mene tietoa, jolloin sille ei tehdä mitään. Yksi vaihtoehto olisi siis, että CGI-skripti hoitaisi viestittelyn Sovereignin Planlink-prosessin kanssa.

Myös ominaisuuden testaamista tulisi parantaa. Tällä hetkellä testit ovat melko alkeelliset, eikä kaikkia mahdollisia vaihtoehtoja ja virhetilanteita käydä läpi. Testejä tulisi siis lisätä ja vanhoja tarkentaa. Myös yksikkötestaus olisi syytä aloittaa.

6.3 Tietoturvan parantaminen

Romexiksen tietoturvaa voitaisiin myös parantaa monella tavalla useammassa eri asetustensiirtoon liittyvässä rajapinnassa. Romexis ja Sovereign keskustelevat socketin kautta TCP-yhteyden yli. Kulkevat viestit ovat suurimmalta osin selkokielisessä tekstimuodossa. Yhteys ei vaadi mitään tunnistautumista, eikä yhteys ole salattu. Näin kulkevaa viestiliikennettä on äärimmäisen helppo seurata. Tällä hetkellä tämä ei ole kovin suuri ongelma, koska laitteiden välillä ei kulje mitään kriittistä dataa. Jos haluttaisiin alkaa siirtää esimerkiksi potilastietoja, tulisi yhteys salata ja vaatia asiakasohjelmilta tunnistautumista. Esimerkiksi SSH:n tai SSL:n käyttöön siirtyminen olisi vaihtoehto.

Myös Romexiksen tietokantayhteyden tietoturvassa on parantamisen varaa. Tietokanta on kyllä suojattu salasanalla, mutta tietokannan lukuun vaadittava salasana löytyy tekstitiedostosta ohjelman asennuskansiosta ja kirjoittamiseen vaadittavan salasanan murtaminen ei liene myöskään kovin vaikeaa. Jälleen kerran tietokannan muokkaaminen voi tuskin kuitenkaan johtaa vaikkapa potilasturvan vaarantumiseen. Sovereignilta saatavat asetustiedostot yksinkertaisesti tallennetaan Romexis-palvelimen kiintolevylle, jolloin niitä pääsee vaivatta muokkaamaan. Jos tietokonet-

ta, jolla Romexis-palvelinta ajetaan, ei ole asianmukaisesti suojattu, voi joku päästessään muuttamaan asetustiedoston sopivia rivejä ehkä aiheuttaa jopa vaaratilanteita. Näitä voisi olla esimerkiksi se, että jonkin instrumentin pyörimisnopeus on väärä, jokin lämpötila väärä tai jokin automaattiasento liikuttaa tuolin vaaralliseen asentoon kriittisellä hetkellä. Asetustiedostot voisi siirtää tietokantaan kunnollisen tunnistautumisen taakse suojaan.

Näiden lisäksi voi Sovereignin Unix-käyttöjärjestelmään kirjautua sisään Telnet-yhteydellä pelkän käyttäjätunnuksen avulla, jolloin ainakin vahinkoa laitteelle on helppo aiheuttaa esimerkiksi tuhoamalla tiedostoja. Tähän olisi helppo parannus siirtyä käyttämään salattua ja salasanalla tai avaimella suojattua SSH-yhteyttä.

Myös asetustiedostojen siirtämiseen käytettävä HTTP-palvelin on suojattu vain yksinkertaisella käyttäjätunnus-salasana-yhdistelmällä. Nämä lienevät helposti murrettavissa tai koodista ongittavissa. Voisi siirtyä käyttämään `basic`-tyyppisen kirjautumisen sijaan parempaa `digest`-tyyppistä kirjautumista. Myös HTTPS-yhteyden käyttäminen SSL:n avulla olisi vaihtoehto. [53] [54] [55] [9]

Kaiken kaikkiaan nämä ovat kuitenkin pieniä puutteita, joita hyödyntämällä pystytään lähinnä haitantekoon. Todellisten vaaratilanteiden aiheuttaminen on hankalampaa ja epätodennäköisempää.

Viitteet

[1] "Planmeca Group", URL http://www.planmeca.com/Company/Planmeca-Group/ (29.10.2013)

[2] J. Kannonkerä, *Käytettävyystekniikkaprosessin soveltaminen lääketieteellisten laitteiden valmistajalle*, Diplomityö, Aalto-yliopiston Teknillinen korkeakoulu (2010)

[3] "Planmeca Sovereign myyntiesite - SGN_bro_fi_0911_low.pdf", URL http://materialbank.planmeca.com (30.5.2013)

[4] J. Sutinen, *Modulaarisen hammashoitokoneen kehittäminen*, Diplomityö, Oulun yliopisto (2007)

[5] Useita kirjoittajia, *Planmeca Romexis User's Manual*

[6] nyholku, "Planmeca PMUAPI Architecture", Planmecan sisäinen dokumentti (4.10.2000)

[7] nyholku, "Planmeca PMUAPI Architecture Changes", Planmecan sisäinen dokumentti (13.11.2006)

[8] D. E. Comer, *Internetworking With TCP/IP*, Osa 1: Principles, Protocols and Architecture, Alan Apt, 4. painos (2000)

[9] J. F. Kurose, K. W. Ross, *Computer Networking A Top-Down Approach Featuring the Internet*

[10] "Internet protocol suite", URL http://en.wikipedia.org/wiki/Internet_protocol_suite (29.10.2013)

[11] "Internet Protocol", URL http://en.wikipedia.org/wiki/Internet_Protocol (29.10.2013)

[12] "RFC 791: Internet Protocol", URL http://tools.ietf.org/html/rfc791 (29.10.2013)

[13] "Transmission Control Protocol", URL http://en.wikipedia.org/wiki/Transmission_Control_Protocol (29.10.2013)

[14] "RFC 793: Transmission Control Protocol", URL http://tools.ietf.org/html/rfc793 (29.10.2013)

[15] "User Datagram Protocol", URL http://en.wikipedia.org/wiki/User_Datagram_Protocol (29.10.2013)

[16] "RFC 768: User Datagram Protocol", URL http://tools.ietf.org/html/rfc768 (29.10.2013)

[17] "Telnet", URL http://en.wikipedia.org/wiki/Telnet (29.10.2013)

[18] "RFC 854: Telnet Protocol Specification", URL http://tools.ietf.org/html/rfc854 (29.10.2013)

[19] "Secure Shell", URL http://en.wikipedia.org/wiki/Secure_Shell (29.10.2013)

[20] "RFC 4251: The Secure Shell (SSH) Protocol Architecture", URL http://tools.ietf.org/html/rfc4251 (29.10.2013)

[21] "Trivial File Transfer Protocol", URL http://en.wikipedia.org/wiki/Tftp (29.10.2013)

[22] "RFC 1350: THE TFTP PROTOCOL (REVISION 2)", URL http://tools.ietf.org/html/rfc1350 (29.10.2013)

[23] "File Transfer Protocol", URL http://en.wikipedia.org/wiki/Ftp (29.10.2013)

[24] "RFC 959: FILE TRANSFER PROTOCOL (FTP)", URL http://tools.ietf.org/html/rfc959 (29.10.2013)

[25] "RFC 2228: FTP Security Extensions", URL http://tools.ietf.org/html/rfc2228 (29.10.2013)

[26] "FTPS", URL http://en.wikipedia.org/wiki/Ftps (29.10.2013)

[27] "RFC 4217: Securing FTP with TLS", URL http://tools.ietf.org/html/rfc4217 (29.10.2013)

[28] "FTP over SSH", URL http://en.wikipedia.org/wiki/FTP_over_SSH#FTP_over_SSH_.28not_SFTP.29 (29.10.2013)

[29] "Secure copy", URL http://en.wikipedia.org/wiki/Secure_copy (29.10.2013)

[30] "SSH File Transfer Protocol", URL http://en.wikipedia.org/wiki/SSH_File_Transfer_Protocol (29.10.2013)

[31] "Network File System", URL http://en.wikipedia.org/wiki/Network_File_System (29.10.2013)

[32] "RFC 4217: Network File System (NFS) version 4 Protocol", URL http://tools.ietf.org/html/rfc3530 (29.10.2013)

[33] "Uniform resource locator", URL http://en.wikipedia.org/wiki/Url (29.10.2013)

[34] S. Guelich, S. Gundavaram, G. Birznieks, *CGI Programming with Perl*, O'Reilly & Associates, Inc., 2. painos (2000)

[35] "Internet media type", URL http://en.wikipedia.org/wiki/MIME_type (29.10.2013)

[36] "RFC 2046: Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", URL http://tools.ietf.org/html/rfc2046 (29.10.2013)

[37] "Hypertext Transfer Protocol", URL http://en.wikipedia.org/wiki/Http (29.10.2013)

[38] "RFC 2616: Hypertext Transfer Protocol – HTTP/1.1", URL http://tools.ietf.org/html/rfc2616 (29.10.2013)

[39] "Transport Layer Security", URL http://en.wikipedia.org/wiki/Transport_Layer_Security (29.10.2013)

[40] "RFC 2246: The TLS Protocol", URL http://tools.ietf.org/html/rfc2246 (29.10.2013)

[41] "IPsec", URL http://en.wikipedia.org/wiki/Ipsec (29.10.2013)

[42] "Introduction to GNU Cgicc", URL http://www.gnu.org/software/cgicc/ (29.10.2013)

[43] "Java Remote Method Invocation", URL http://en.wikipedia.org/wiki/Java_RMI (29.10.2013)

[44] P. Finnilä, "ACCU Software Design Specification", Planmecan sisäinen dokumentti (13.7.2012)

[45] "ACCU Operator Process Design Specification", Planmecan sisäinen dokumentti

[46] "ACCU:n tilakaaviot", Planmecan sisäinen dokumentti

[47] "ACCU:n viestirajapinnat", Planmecan sisäinen dokumentti

[48] "Romexis Architecture", Planmecan sisäinen dokumentti

[49] J. Räikkönen, "PLANLINK and PMUAPI", Planmecan sisäinen dokumentti (5.5.2011)

[50] "Sovereignin asetusmalli", Planmecan sisäinen dokumentti (27.11.2009)

[51] "Sovereignin oletusasetukset", Planmecan sisäinen dokumentti (10.11.2010)

[52] A. Wallenius, "Toteutusehdotus", Planmecan sisäinen dokumentti (2013)

[53] "Basic access authentication", URL http://en.wikipedia.org/wiki/Basic_access_authentication (29.10.2013)

[54] "Digest access authentication", URL http://en.wikipedia.org/wiki/Digest_access_authentication (29.10.2013)

[55] "RFC 2617: HTTP Authentication: Basic and Digest Access Authentication", URL https://www.ietf.org/rfc/rfc2617.txt (29.10.2013)

[56] M. Benantar, *Access Control Systems - Security, Identity Management and Trust Models*, Springer Science+Business Media, 1. painos (2006)

[57] *INTERNATIONAL STANDARD IEC 80001-1 Application of risk management for IT-networks incorporating medical devices*, Osa 1: Roles, responsibilities and activities, International Electrotechnical Commission (IEC), 1. painos (2010)

[58] I. Pöyhönen, K. Kylmälä, *Terveydenhuollon laadunhallinta - Lääkintälaitejärjestelmien turvallisuus*, Lääkelaitos (2004)

[59] A. Wallenius, "Sovereign Settings Transfer", Planmecan sisäinen dokumentti (2013)